TUGAS AKHIR – TF 141581

RANCANG BANGUN SISTEM DETEKSI GARIS MARKA JALAN DENGAN METODE *HOUGH TRANSFORM* BERBASIS *RASPBERRY PI*

TAREZQI MOCHTAR ROHMA
NRP. 02311440000047

Dosen Pembimbing :
Ir. Apriani Kusumawardhani, MSc
Andi Rahmadiansah, ST. MT.

DEPARTEMEN TEKNIK FISIKA
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2018

Halaman Ini Sengaja Dikosongkan

FINAL PROJECT – **TF 141581**

DESIGN SYSTEM DETECTION OF ROAD MARKINGS WITH HOUGH TRANSFORM METHOD BASED RASPBERRY PI

TAREZQI NOCHTAR ROHMA
NRP. 02311440000047

Supervisors :
Ir. Apriani Kusumawardhani, MSc
Andi Rahmadiansah, ST. MT.

ENGINEERING PHYSICS DEPARTMENT
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya 2018

Halaman Ini Sengaja Dikosongkan

PERNYATAAN BEBAS PLAGIASME

Saya yang bertanda tangan di bawah ini

Nama	: Tarezqi Mochtar Rohma
NRP	: 02311440000047
Jurusan/ Prodi	: Teknik Fisika/ S1 Teknik Fisika
Fakultas	: Fakultas Teknologi Industri
Perguruan Tinggi	: Institut Teknologi Sepuluh Nopember

Dengan ini menyatakan bahwa Tugas Akhir dengan judul "Rancang Bangun Sistem Deteksi Garis Marka Jalan Dengan Metode *Hough Transform* Berbasis *Raspberry PI*" adalah benar karya saya sendiri dan bukan plagiat dari karya orang lain. Apabila di kemudian hari terbukti terdapat plagiat pada Tugas Akhir ini, maka saya bersedia menerima sanksi sesuai ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat dengan sebenar-benarnya.

Surabaya, 10 Juni 2018
Yang membuat pernyataan,

Tarezqi Mochtar R.
NRP. 02311440000047

Halaman Ini Sengaja Dikosongkan

LEMBAR PENGESAHAN I

TUGAS AKHIR

RANCANG BANGUN SISTEM DETEKSI GARIS MARKA JALAN DENGAN METODE HOUGH TRANSFORM BERBASIS RASPBERRY PI

Oleh:

Tarezqi Mochtar Rohma
NRP 02311440000047

Surabaya, 22 April 2018

Menyetujui,
Dosen Pembimbing I

Ir. Apriani Kusumawardhani, MSc
NIPN. 19530404 197901 2 001

Menyetujui,
Dosen Pembimbing II

Andi Rahmadiansah, ST. MT.
NIPN. 19790517 200312 1 002

Mengetahui,
Kepala Departemen
Teknik Fisika FTI-ITS

Agus Muhamad Hatta, S.T., M.Si, Ph.D
NIPN. 19780902 200312 1 002

Halaman Ini Sengaja Dikosongkan

LEMBAR PENGESAHAN II
RANCANG BANGUN SISTEM DETEKSI GARIS MARKA JALAN DENGAN METODE HOUGH TRANSFORM BERBASIS RASPBERRY PI

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memeperoleh Gelar Sarjana Teknik
pada
Bidang Studi Rekayasa Fotonika
Progam Studi S-1 Departemen Teknik Fisika
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember

Oleh:

Tarezqi Mochtar Rohma
NRP. 02311440000047

Disetujui oleh Tim Penguji Tugas Akhir:

1. Ir.Apriani Kusumawardhani,MSc.....(Pembimbing I)
2. Andi Rahmadiansah, ST. MT.....(Pembimbing II)
3. Agus Muhamad Hatta, S.T., M.Si, Ph.D.....(Penguji I)
4. Prof. Dr. Ir. Sekartedjo, M.Sc.....(Penguji II)
5. Lizda Johar Mawarani,ST. MT.....(Penguji III)

SURABAYA
JUNI, 2018

Halaman Ini Sengaja Dikosongkan

RANCANG BANGUN SISTEM DETEKSI GARIS MARKA JALAN DENGAN METODE HOUGH TRANSFORM BERBASIS RASPBERRY PI

Nama	**: Tarezqi Mochtar Rohma**
NRP	**: 02311440000047**
Departemen	**: Teknik Fisika FTI-ITS**
Dosen Pembimbing	**: 1. Ir. Apriani Kusumawardhani, MSc**
	2. Andi Rahmadiansah, ST. MT.

Abstrak

Marka Jalan adalah suatu tanda yang berada di permukaan jalan yang meliputi peralatan atau tanda yang membentuk garis membujur, garis melintang, garis serong, serta lambang yang berfungsi untuk mengarahkan arus lalu lintas dan membatasi daerah kepentingan lalu lintas. Pada era modern ini, terutama pada tahun 2018 di dunia mulai menggencarkan produk otomotif terbaru tentang sistem automasi yang diterapkan pada mobil kendaraan. Berbagai riset dilakukan untuk menghasilkan suatu kendaraan *Self Driving Car* untuk memberikan kenyamanan, keamanan, dan kepuasan pengemudi kendaraan. Sehingga dirancang suatu sistem deteksi marka jalan dengan metode *Hough Transform* berbasis *Raspberry Pi.* Pada tugas akhir ini, dilakukan deteksi pada marka membujur yaitu marka lurus utuh, marka lurus putus, marka membelok kekiri dan membelok kekanan. Informasi atau parameter tambahan yang dimasukkan dalam sistem ini adalah besar jari–jari kelengkungan, posisi kendaraan terhadap marka tengah dan marka samping, dan keterangan kendaraan apakah kendaraan boleh mendahului pengendara lain atau tidak. Purwarupa marka yang telah dibuat dengan rasio 16 kali lebih kecil dari ukuran sebenarnya sebagai acuan ukuran purwarupa mobil kendaraan. Penempatan ukuran marka jalan dan pembuatan mengikuti aturan peraturan Menteri Perhubungan Republik Indonesia Nomor 34 tahun 2014 tentang Marka Jalan. Hasil uji coba sistem rancang bangun ini dengan menggunakan dua variasi kecepatan yaitu sistem deteksi marka dapat mendeteksi dengan baik pada kecepatan pertama yaitu 74,4 cm/s dengan rata-rata *error*

pada marka belok kiri 11,38%, marka belok kanan 13,45%, marka lurus putus 6,4% dan lurus utuh 6,51%. Sistem deteksi marka kurang dapat mendeteksi dengan baik pada kecepatan kedua yaitu 115,3 cm/s dengan rata-rata *error* pada marka belok kiri 26,23%, marka belok kanan 23,25%. Sedangkan untuk marka lurus memiliki deteksi sangat baik pada kecepatan kedua karena untuk perhitungan kelengkungan marka tidak diperhitungkan. Nilai rata-rata marka lurus putus pada kecepatan kedua adalah 6,72% dan lurus utuh 6,95%.

Kata Kunci : Marka Jalan, Hough Transform, Raspberry Pi

DESIGN SYSTEM DETECTION OF ROAD MARKINGS WITH HOUGH TRANSFORM METHOD BASED RASPBERRY PI

Name	**: Tarezqi Mochtar Rohma**
NRP	**: 02311440000047**
Department	**: Engineering Physics FTI-ITS**
Supervysors	**: 1. Ir. Apriani Kusumawardhani, MSc**
	2. Andi Rahmadiansah, ST. MT.

Abstract

Road marks are a sign on the road surface that includes equipment or marks that form longitudinal lines, transverse lines, oblique lines, and symbols that serve to direct the flow of traffic and limit the area of interest of traffic. In this modern era, especially in 2018 in the world began to intensify the latest automotive products about the automation system applied to vehicle cars. Various researches were conducted to produce a Self Driving Car vehicle to provide the comfort, safety, and satisfaction of the driver of the vehicle. So, on this experiment, a road marking detection system with Raspberry Pi-based Hough Transform method designed. In this final project, detection on the longitudinal marker that is a straight line intact, straight cut mark, turn left and turn right had been done. Additional information or parameters included in this system are the large radius of curvature, the position of the vehicle against the middle mark and the side mark, and the vehicle's description whether the vehicle may precede other riders or not. The markup prototype has been made with a ratio 16 times smaller than the actual size as the prototype size of the vehicle's car. The placement of road and building mark sizes follows the rules of the Minister of Transportation of the Republic of Indonesia No. 34 of 2014 on Road Marking. The test results of this design system using two

variations of speed that is the detection system markers can detect well at the first speed of 74.4 cm / s with the average error on the left turning mark 11,38%, marka turn right 13,45 %, straight cut marks 6.4% and straight full 6.51% intact. The marker detection system is less able to detect well at the first velocity of 115,3 cm / s with an average error on the left turn mark of 26,23%, 23,25% right turn mark. As for the straight marker has a very good detection at the second speed because for the calculation of the curvature of the mark is not taken into account. The average value of the straight cut marks at the second velocity is 6,72% and is intact 6.95% intact.

Keywords : Road Mark, Hough transform, Raspberry Pi

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang senantiasa melimpahkan rahmat serta hidayah-Nya, serta shalawat serta salam kepada Nabi Muhammad SAW, hingga terselesaikannya Tugas Akhir beserta Laporan Tugas Akhir yang berjudul **RANCANG BANGUN SISTEM DETEKSI GARIS MARKA JALAN DENGAN METODE HOUGH TRANSFORM BERBASIS RASPBERRY PI.**

Penulis telah banyak memperoleh bantuan dari berbagai pihak dalam penyelesaian Tugas Akhir dan laporan Tugas Akhir ini. Penulis mengucapkan terimakasih kepada :

1. Bapak Agus Muhamad Hatta, ST, MSi, Ph.D selaku Ketua Departemen Teknik Fisika dan dosen wali yang telah memberikan petunjuk, ilmu, serta bimbingan selama menempuh pendidikan di Teknik Fisika.
2. Bapak Hendra Cordova ST.MT. selaku Ketua Prodi S1 dan dosen wali penulis yang telah membimbing selama perkuliahan.
3. Kedua orang tua serta saudara terimakasih atas segala cinta, kasih sayang, doa, perhatian, serta dukungan moril dan materiil yang telah diberikan.
4. Ibu Ir. Apriani Kusumawardhani, MSc dan Bapak Andi Rahmadiansah, ST. MT. selaku dosen pembimbing yang telah dengan sabar memberikan petunjuk, ilmu, serta bimbingan yang sangat bermanfaat.
5. Bapak Prof. Dr. Ir. Sekartedjo, MSc. selaku Kepala Laboratorium Rekayasa Fotonika yang telah memberikan ilmu, petunjuk, nasihat, serta kemudahan perizinan.
6. Seluruh teman Tugas Akhir (Moh. Fiqih, Roni dkk), terima kasih untuk semuanya.
7. Seluruh dosen, karyawan dan civitas akademik Teknik Fisika FTI-ITS, terimakasih atas segala bantuan dan kerjasamanya.

Penulis sadar bahwa penulisan laporan Tugas Akhir ini tidaklah sempurna, namun semoga laporan ini dapat memberikan kontribusi yang berarti dan menambah wawasan yang bermanfaat bagi pembaca, keluarga besar Teknik Fisika khususnya, dan civitas akademik ITS pada umumnya. Selain itu juga semoga dapat bermanfaat sebagai referensi pengerjaan laporan Tugas Akhir bagi mahasiswa yang lain.

Surabaya, 10 Juni 2018

Penulis

DAFTAR ISI

DAFTAR GAMBAR

Halaman Ini Sengaja Dikosongkan

DAFTAR TABEL

Halaman Ini Sengaja Dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

Marka Jalan adalah suatu tanda yang berada di permukaan jalan atau di atas permukaan jalan yang meliputi peralatan atau tanda yang membentuk garis membujur, garis melintang, garis serong, serta lambang yang berfungsi untuk mengarahkan arus lalu lintas dan membatasi daerah kepentingan lalu lintas. Seiring perkembangan teknologi dan ilmu pengetahuan yang semakin pesat, mendorong manusia untuk menciptakan alat yang dapat memberikan efisiensi (biaya dan waktu), keamanan dan kenyamanan yang tinggi. Hal tersebut juga berlaku diseluruh aspek terutama di bidang lalu lintas. Pada era modern ini, terutama pada tahun 2018 di dunia mulai menggecarkan produk otomotif terbaru tentang sistem automasi yang diterapkan kepada mobil kendaraan. Beberapa perusahaan besar seperti Google dan perusahaan otomotif seperti Waymo telah resmi meluncurkan produk *Self Driving Car* yang beroprasi di jalan raya [1] [2]. Salah satu hasil survey resmi internasional pada Statista, tahun 2016 tentang penggunaan sistem *autonomus car* di dunia semakin meningkat. Beberapa negara yang telah didata dalam survey tersebut diantaranya adalah India, China, Amerika Serikat, Inggris, Perancis, Jerman dan Jepang [1]. Pada nilai keseluruhan data tersebut menunjukan bahwa di India memiliki presentase tertinggi yaitu 56% memilih untuk setuju menggunakan Self Driving Car dalam kehidupan sehari-hari dan hanya 3% memilih untuk tidak. Sedangkan untuk negara Jepang memiliki presentase terendah yaitu 12% memilih untuk tidak menggunakan *Self Driving Car* dalam kehidupan sehari-hari dan 22% memilih untuk tidak [1]. Berbagai riset dilakukan untuk menghasilkan suatu kendaraan *Self Driving Car* untuk memberikan kenyamanan, keamanan dan kepuasan pengemudi kendaraan.

Akan tetapi, mobil dengan spesifikasi *Self Driving* tersebut memiliki harga yang relatif mahal dan kurangnya daya saing terutama di Indonesia. Sehingga dengan dijalankannya tugas akhir ini yaitu pembuatan purwarupa deteksi ini, diharapkan Indonesia

mampu merancang dan membangun sendiri produk *Self Driving Car* yang mampu bersaing dengan produk di dunia dan memiliki harga yang cukup dikalangan warga Indonesia.

Pada tugas akhir ini akan direalisasikan Rancang Bangun Sistem Deteksi Garis Marka Jalan dengan Metode *Hough Transform* Berbasis *Raspberry Pi*. Alat ini berfungsi untuk mendeteksi garis atau marka jalan secara realtime dimana hasil keluaran dari deteksi tersebut berupa parameter berupa deteksi marka jalan, nilai jari – jari kelengkungan marka pada tikungan, dan posisi kendaraan terhadap garis marka. Alat ini menggunakan masukan berupa citra gambar pengambilan video secara *realtime* dari sensor kamera. Kamera menangkap citra garis atau marka jalan, selanjutnya gambar akan diproses dengan *microcontroller* berupa *Raspberry Pi 2* untuk mengetahui parameter pendektesian yang telah disebutkan diatas. posisi kendaraan terhadap marka jalan untuk berada pada jalur yang benar. Jika kendaraan memasuki jalur yang salah, secara automatis keluaran dari hasil pengolahan citra pada *Raspberry Pi* berupa parameter posisi berupa angka dan penunjukan posisi kendaraan terhadap marka. Dengan sistem otomatis ini diharapkan Indonesia mampu "mengikuti jaman teknologi mengenai *Self Driving Car* yang aman dan baik yang menjadi daya saing modern didunia otomotif.

1.2 Rumusan Masalah

Dari latar belakang yang telah diuraikan, maka didapatkan permasalahan dalam tugas akhir ini adalah Bagaimana merancang sistem untuk mendeteksi garis Marka jalan secara *realtime* berbasis *image prossesing* menggunakan metode *Hough Transform* dan mengimplementasikan metode *Hough Transform* kedalam *Microcontroller Raspberry Pi*?

1.3 Tujuan

Tujuan yang ingin dicapai dari tugas akhir ini yaitu merancang sistem untuk mendeteksi garis Marka jalan secara *realtime* berbasis *image prossesing* menggunakan metode *Hough Transform* dan

mengimplementasikan metode *Hough Transform* kedalam *Microcontroller Raspberry Pi*.

1.4 Batasan Masalah

Dalam proses Rancang Bangun Sistem Autonomus Car Berdasarkan Garis Marka Jalan Berbasis Raspberry Pi, perlu adanya batasan masalah agar permasalahan yang diterangkan pada sub-bab tidak meluas, antara lain:

- Pendektesian Marka jalan *secara Real-time* menggunakan purwarupa marka dari marka sebenarnya dengan rasio 1:16 dengan dua sisi satu arah dengan type marka lurus dan marka dengan jari – jari kelengkungan 55 cm.
- Pendektesian Marka jalan secara *Real-time* menggunakan metode *Hough Transform* dalam menentukan garis Marka jalan purwarupa
- Uji coba menggunakan *microcontroller Raspberry Pi* sebagai basis program data
- Menggunakan spesifikasi kamera Pi versi 2.0 dengan resolusi 5 Megapixel dan maksimal *Frame Per Second* hanya 30 detik

1.5 Sistematika Laporan

Sistematika penulisan laporan tugas akhir adalah sebagai berikut:

BAB I Pendahuluan

Bab I ini terdiri dari latar belakang, perumusan masalah, tujuan, batasan masalah dan sistematika laporan.

BAB II Teori Penunjang

Bab II ini dibahas mengenai teori-teori yang berkaitan dengan penelitian yang akan dilakukan, seperti pemahaman mengenai Raspberry Pi, tahab pemprosesan citra gambar, metode *Hough Transform*, dasar ilmu tentang jari – jari kelengkungan, dan pengetahuan tentang Marka jalan.

BAB III Metodologi Penelitian

Bab ini berisi mengenai rancangan dari penelitian yang dilakukan, metode, dan langkah-langkah dalam penelitian.

BAB IV Analisis Data dan Pembahasan

Bab ini berisi tentang data hasil penelitian dari deteksi rancang bangun marka jalan menggunakan Raspberry Pi dengan metode *Hough Transform* dan analisis dari performansi sistem deteksi marka dan parameter penunjang yaitu jari – jari kelengkungan dan posisi rancangan bangun terhadap marka jalan.

BAB V Kesimpulan dan Saran

Bab ini diberikan kesimpulan tentang tugas akhir yang telah dilakukan berdasarkan data-data yang diperoleh, serta diberikan saran sebagai penunjang maupun pengembangan tugas akhir selanjutnya.

BAB II
TINJAUAN PUSTAKA

Pada bab ini selanjutnya dijelaskan mengenai tinjauan pustaka apa saja yang mendasari penelitian ini.

2.1 *Raspberry Pi*

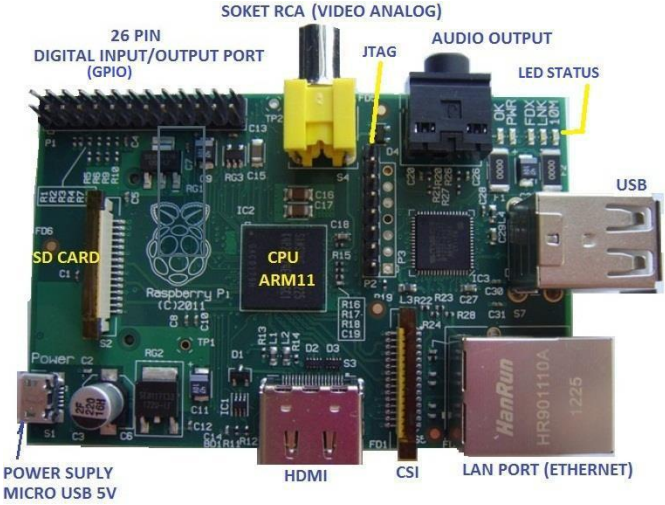
Raspberry Pi, sering disingkat dengan nama Raspi, adalah komputer papan tunggal (single-board circuit; SBC) yang seukuran dengan kartu kredit yang dapat digunakan untuk menjalankan program perkantoran, permainan komputer, dan sebagai pemutar media hingga video beresolusi tinggi. Raspberry Pi dikembangkan oleh yayasan nirlaba yaitu Rasberry Pi Foundation, yang digawangi sejumlah pengembang dan ahli komputer dari Universitas Cambridge, Inggris [3]. Serta diilustrasikan pada Gambar 2.1.

2.1.1 Spesifikasi *Raspberry Pi* [4]

Raspberry Pi board dibuat dengan model yang berbeda yaitu *Raspberry Pi type A ,A+ Raspberry Pi type B.,B+ Raspberry pi 2,Rasberry pi 3,Raspberry Pi zero*. Perbedaannya antara lain pada RAM*(Random Access Memory)* dan *Port* LAN. Type A memiliki kapasitas RAM = 256 Mb dan tanpa *port* LAN(*ethernet*), type B memiliki kapasitas RAM = 512 Mb dan terpasang *port* untuk LAN. *Raspberry Pi* board mempunyai *input* dan *output* antara lain :

- HDMI, dihubungkan ke LCD TV atau monitor PC yang mempunyai port HDMI.
- *Audio output*
- 4 buah *port* USB digunakan untuk *keyboard* dan *mouse*
- 26 pin I/O *digital*
- CSI port (*Camera Serial Interface*)
- DSI (*Display Serial Interface*)
- LAN port (*network*)
- SD Card slot untuk SD Card memori yg menyimpan sistem operasi berfungsi seperti *hardisk* pada PC.
- *Power Supply Micro USB 5V*

Hasil ilustrasi mengenai Raspberry Pi terdapat pada gambar 2.1 dibawah ini.



Gambar 2. 1 Model Raspberry Pi 2 **[5]**

2.1.2 Pin Port GPIO

GPIO merupakan sederet pin yang terdiri dari 40 pin dengan berbagai fungsi. Salah satu fitur yang kuat dari *Raspberry Pi* adalah deretan GPIO (tujuan umum *input / output*) pin di sepanjang tepi atas pin board. Sehingga GPIO merupakan antarmuka fisik antara Pi dan perangkat eksternal. Pada tingkat yang paling sederhana, GPIO dapat disebut sebagai *switch* yang diaktifkan atau dinonaktifkan *(input)* atau bahwa Raspberry Pi dapat diaktifkan atau dinonaktifkan *(output)* [5].

Dari 26 pin GPIO yang dimiliki *Raspberry Pi*, terdapat 2 pin sebagai sumber tegangan 5 V, 2 pin sumber tegangan 3.3 V, 5 pin *ground*, 17 pin *input / output*. GPIO pada *Raspberry Pi* dapat dikendalikan dan dipicu dengan berbagai cara, bisa dengan terminal menggunakan *bash script* atau dengan bahasa program

yang lain seperti *Python*. Pin *port* GPIO diilustrasikan pada gambar 2.2 dibawah ini.

Gambar 2. 2 Fungsi *port* GPIO *Raspberry Pi 2* **[5]**

Bagian-bagian port GPIO adalah sebagai berikut:

- Port input output atau masukan pengambilan data dari sensor dan keluaran menuju aktuator.
- Keluaran berupa tegangan pada port 1 dan 2.
- Ground sebagai pengaman
- UART interface sebagai antarmuka UART
- SP10 interface sebagai antarmuka SP10
- 12C interface sebagai antarmukaa 12C

2.2 Pengolahan Citra

Citra adalah istilah lain yang biasa digunakan untuk menggantikan kata gambar, dan memegang peranan penting sebagai salah satu media informasi berupa informasi visual. Selain itu dari citra dapat diartikan merupakan gambar pada bidang dua dimensi dan dapat juga diartikan citra merupakan fungsi menerus dari intensitas cahaya pada bidang dua dimensi. Sumber cahaya akan membuat objek diterangi dan objek akan memantulkan kembali sebagian dari berkas cahaya tersebut. Pantulan ini akan ditangkap kembali oleh alat-alat optik, misalnya ketika membaca buku mata akan dapat melihat jika ada sumber cahaya yang menerangi buku. Contoh alat-alat optik yang biasa digunakan, misalnya mata manusia, kamera, pemindai, dan sebagainya [6].

Pengolahan citra adalah suatu pemrosesan citra sehingga menghasilkan citra yang sesuai yang diinginkan atau memiliki kualitas yang lebih baik. Sebuah citra terkadang memiliki penurunan informasi seperti mengandung derau, kontras yang kurang sesuai, kurang tajam, kabur, dan sebagainya. Sehingga diperlukan adanya manipulasi gambar sehingga kualitasnya lebih baik. Selain itu pengolahan citra bertujuan untuk mudah menginterpretasi citra oleh manusia maupun mesin (komputer). Sehingga terdapat banyak teknik yang digunakan untuk mengolahan citra sehingga dapat ditransformasikan citra menjadi citra lain.

2.2.1 Derajat keabuan

Derajat keabuan adalah salah satu format citra yang digunakan dalam pemrosesan citra digital. Format citra ini berupa skala keabuan yang memiliki nilai 0 hingga 225 dimana 0 bernilai gelap dan 225 bernilai putih [6].

Derajat keabuan memiliki beberapa nilai dan nilainya tergantung dengan nilai kedalaman pixel yang dimiliki oleh citra. Berikut beberapa pembagian nilai derajat keabuan yang berhubungan dengan kedalaman pixel pada tabel dibawah ini.

Tabel 2.1 Perbandingan Derajat Keabu-abuan

Gray scale	**Skala**	**Kedalaman pixel**
2	0 dan 1	1 bit
4	0 hingga 3	2 bit
16	0 hingga 15	4 bit
256	0 hingga 255	8 bit

2.2.2 Pengaburan Citra

Pengaburan citra atau yang sering disebut dengan Smoothing atau Blurring adalah metode yang digunakan dalam pengolahan citra untuk mengaburkan objek.

Tujuan dari mengaburkan objek ini adalah untuk mengurangi derau pada citra sehingga mempermudah pemrosesan citra. Pengaburan citra sangat berguna ketika proses pendektesian tepi [7].

2.2.3 Parser

Istilah parsing berasal dari bahasa Latin *pars* (*orationis*), yang berarti bagian (bicara) . Parse, Sintaksis atau analisis sintaksis adalah komponen perangkat lunak yang mengambil data input

(sering teks) dan membangun struktur data berupa semacam pohon parse , pohon sintaksis abstrak atau struktur hierarkis lainnya, memberikan representasi struktural dari input saat memeriksa sintaks yang benar [8].

Dalam linguistik komputasional istilah ini digunakan untuk merujuk pada analisis formal oleh komputer kalimat atau rangkaian kata lain ke dalam konstituennya, menghasilkan pohon parse yang menunjukkan hubungan sintaksisnya satu sama lain, yang mungkin juga mengandung informasi semantik dan lainnya.

2.2.4 Deteksi Tepi *Canny*

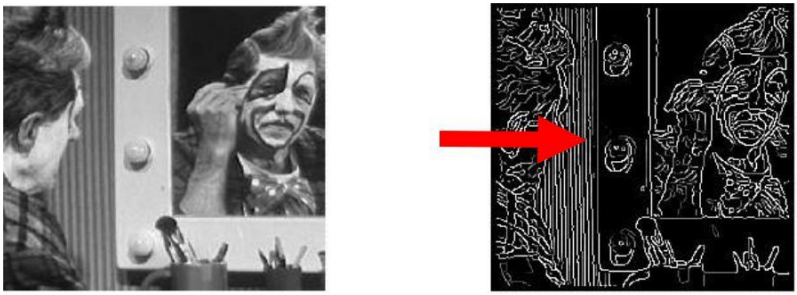
Canny edge detection merupakan salah satu metode deteksi tepi yang cukup populer penggunaannya dalam pengolahan citra. Salah satu alasannya adalah ketebalan tepi yang bernilai satu pixel yang dimaksudkan untuk melokalisasi posisi tepi pada citra secara sepresisi mungkin [9]. Algoritma *canny edge detection* secara umum beroperasi sebagai berikut :

- Penghalusan untuk mengurangi dampak *noise* terhadap pendeteksian edge. Biasanya teknik yang digunakan pada tahap ini adalah *Gaussian Blur*. Proses *Gaussian Blur* dapat dilakukan terhadap citra secara keseluruhan (hasil akhir berupa 1 citra baru), atau dilakukan terpisah (hasil akhir berupa dua buah citra yaitu blur horizontal dan vertikal). Hasil dari *Gaussian Blur* akan digunakan dalam langkah selanjutnya yaitu menentukan potensi gradien citra.
- Menghitung potensi gradien citra. Pada langkah menghitung potensi gradien citra ada dua buah informasi yang dibutuhkan yaitu kekuatan tepi (*edge strength/magnitude*), dan arah tepi (*edge direction/orientation*). Operator sobel memanfaatkan dua buah template edge pada dua arah tegak lurus (horizontal dan vertikal) dan menghitung arah tepi dari *arctangent* kedua nilai tersebut.
- *non-maximal supression* dari gradien citra untuk melokalisasi tepi secara presisi. Hasil penerapan operator gradien untuk menghitung potensi gradien di tahap sebelumnya tidak

memberi informasi secara spesifik tentang lokasi dari tepi yang dicari. Alternatifnya adalah menggunakan operator *zero-crossing* yang digunakan oleh algoritma deteksi *Marr-Hildreth*. *Non-maximal supression* bertujuan membuang potensi gradien di suatu piksel dari kandidat edge jika piksel tersebut bukan merupakan maksimal lokal pada arah tepi di posisi piksel tersebut

- *hysteresis thresholding* untuk melakukan klasifikasi akhir. Langkah terakhir adalah *thresholding* atau klasifikasi tiap piksel apakah termasuk dalam kategori piksel *edge* atau tidak. Pada tahap ini bisa saja menggunakan *threshold* yang berdasarkan pada satu nilai tertentu. Namun pemilihan *threshold* yang hanya menggunakan satu nilai ini memiliki keterbatasan yaitu adanya kemungkinan piksel yang hilang padahal sebetulnya meruapakan piksel *edge* (*false-negative*) ataupun dimasukkannya piksel yang sebetulnya merupakan *noise* sebagai piksel *edge* (*false-positive*).

Pada gambar 2.3 dibawah ini dijabarkan salah satu contoh mengenai deteksi tepi menggunakan metode *Canny Detection.*



Gambar 2. 3 Contoh deteksi tepi *Canny*

2.2.5 Thresholding

Segmentasi yang paling sederhana dilaksanakan dengan menggunakan ambang intensitas. Nilai yang lebih kecil daripada nilai ambang diperlakukan sebagai area pertama dan yang lebih besar daripada atau sama dengan nilai ambang dikelompokkan sebagai area yang kedua [10].

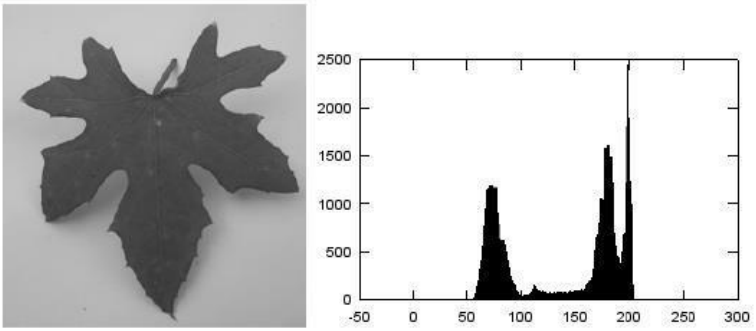
Dalam hal ini, salah satu area tersebut berkedudukan sebagai latar belakang. Cara seperti itulah yang disebut peng-ambangan (bi-level thresholding) atau terkadang dinamakan pengambangan intensitas. Secara matematis, hal itu dinyatakan dengan rumus 2.1 dibawah ini.

$$b(y,x) = f(x) = \begin{cases} 1, untuk\ (y,x) \geq T \\ 0, untuk\ (y,x) < T \end{cases} \quad (2.1)$$

Pada rumus di atas, T menyatakan ambang intensitas. Dalam praktik, nilai 1 atau 0 pada Persamaan 9.1 dapat dipertukarkan. Peng-ambangan intensitas biasa digunakan untuk memisahkan tulisan hitam yang berada di atas secarik kertas putih. Namun, perlu diketahui, pengambangan ini mempunyai kelemahan,yaitu:

- Tidak memperlihatkan hubungan spasial antarpiksel
- Sensitif terhadap pencahayaan yang tidak seragam
- Hanya berlaku untuk keadaan yang ideal (misalnya, latarbelakang hitam dan objek yang bewarna putih).

Pada gambar 2.4 dibawah ini dijabarkan salah satu contoh mengenai *Thresholding.*



(a) ipomoea.png (b) Histogram citra

Gambar 2. 4 Citra daun dan *thresholding* histogramnya

2.2.6 Transformasi *Hough*

Transformasi Hough adalah salah satu metode untuk mendeteksi kurva parametrik seperti garis, lingkaran, ellips, parabola dan lain-lain. Pada awalnya transformasi Hough belum digunakan secara meluas dikarenakan media penyimpanan dan prosesor komputasi yang besar. Tetapi dengan kemajuan perangkat keras komputer dan pengembangan algoritma transformasi Hough permasalahan tersebut dapat diatas [11].

Metode ini juga memiliki efek untuk mengurangi fitur pencarian suatu garis pada gambar asli yang dikembalikan pada transformasi gambar baru. Untuk menggunakan transformasi garis Hough diperlukan adanya proses pendektesian garis beserta proses segmentasi. Prinsip kerja dari transformasi garis Hough adalah dengan mencari bentuk geometri yang paling sesuai dari kumpulan titik [11].

- Transformasi *Hough* Untuk Mendeteksi Garis

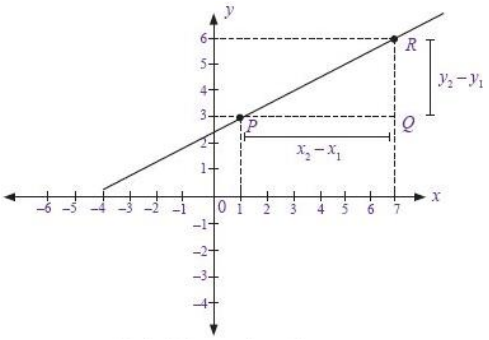
Secara umum persamaan garis lurus dapat dinyatakan dalam bentuk *slope intercept*.

$$y = ax + b \quad (2.2)$$

Dimana a dinyatakan *slope* dan b dinyatakan *intercept*. Karena a dan b dapat bernilai sampai tak hingga maka dapat diatasi dengan menggunakan persamaan kutub.

$$\rho = x \cos\theta + y \sin\theta \quad (2.3)$$

Dengan θ = sudut *relative* kearah horizontal (sumbu x) dan ρ = jarak normal dari origin kearah garis tersebut. Ilustrasi penyajian garis ini dapat disimak pada gambar 2.5 dibawah ini.



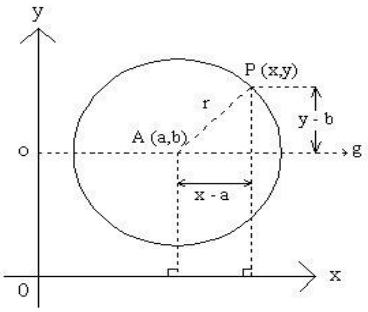
Gambar 2. 5 Penyajian garis *Hough Line*

- Transformasi *Hough* Untuk Mendeteksi Lingkaran

  Secara umum persamaan parametric dari lingkaran dapat didefinisikan sebagai berikut :

$$(x - a)^2 + (y - b)^2 = r^2 \qquad (2.4)$$

Dimana persamaan tersebut mempunyai tiga parameter yaitu *a* dan *b* sebagai pusat lingkaran dan *r* sebagai jari-jari. Ilustrasi penyajian garis ini dapat disimak pada gambar 2.6 dibawah ini.



Gambar 2. 6 Penyajian garis *Hough* lingkaran

2.3 Marka Jalan

Marka Jalan adalah Marka Jalan yang sejajar dengan sumbu jalan. suatu tanda yang berada di permukaan jalan atau di atas permukaan jalan yang meliputi peralatan atau tanda yang membentuk garis membujur, garis melintang, garis serong, serta lambang yang berfungsi untuk mengarahkan arus lalu lintas dan membatasi daerah kepentingan lalu lintas [12]. Marka jalan diatur dalam Peraturan Menteri Perhubungan Nomor 34 tahun 2014.

2.3.1 Marka Jalan Membujur

Marka membujur adalah tanda yang sejajar dengan sumbu jalan [12]. Marka membujur yang dihubungkan dengan garis melintang yang dipergunakan untuk membatasi ruang parkir pada jalur lalu lintas kendaraan, tidak dianggap sebagai Marka jalan

membujur.

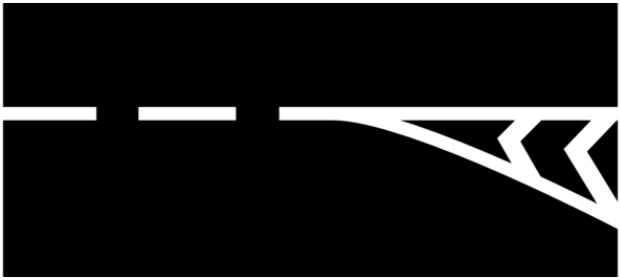
Gambar 2. 7 Marka membujur

Marka membujur harus memiliki ciri – ciri sebagai berikut:

- Marka membujur harus memiliki panjang dengan ukuran yang sama yaitu 3 (tiga) meter, untuk jalan dengan kecepatan rencana kurang dari 60 (enam puluh) kilometer per jam dan 5 (lima) meter, untuk jalan dengan kecepatan rencana 60 (enam puluh) kilometer per jam atau lebih.
- Marka membujur harus memiliki lebar paling sedikit 10 (sepuluh) sentimeter.
- Marka membujur memiliki jarak antar marka 5 (lima) meter, untuk jalan dengan kecepatan rencana kurang dari 60 (enam puluh) kilometer per jam dan 8 (delapan) meter, untuk jalan dengan kecepatan rencana 60 (enam puluh) kilometer per jam atau lebih.

2.3.2 Marka Serong

Marka Serong adalah Marka Jalan yang membentuk garis utuh yang tidak termasuk dalam pengertian Marka Membujur atau Marka Melintang, untuk menyatakan suatu daerah permukaan jalan yang bukan merupakan jalur lalu lintas kendaraan [12].



Gambar 2. 8 Marka serong

Marka Serong berupa garis utuh yang dibatasi dengan rangka garis putus-putus digunakan untuk menyatakan kendaraan tidak boleh memasuki daerah tersebut sampai mendapat kepastian selamat.

2.3.3 Marka Melintang

Marka Melintang berupa garis utuh menyatakan batas berhenti kendaraan yang diwajibkan berhenti oleh alat pemberi isyarat lalu lintas, rambu berhenti, tempat penyeberangan, atau zebra cross [12].

Gambar 2. 9 Marka melintang

Marka Melintang berupa garis putus-putus berfungsi untuk menyatakan batas yang tidak dapat dilampaui kendaraan sewaktu memberi kesempatan kepada kendaraan yang mendapat hak utama pada persimpangan.

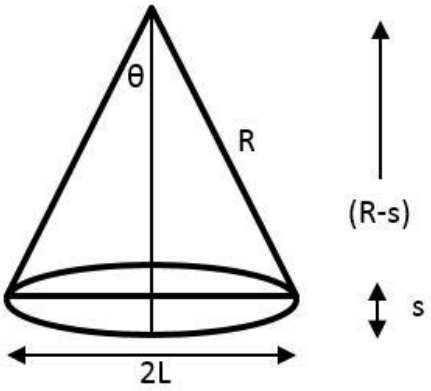
2.4 Jari – Jari Kelengkungan

Radius atau jari-jari sebuah lingkaran adalah garis yang menghubungkan titik pusat lingkaran dengan satu titik pada lingkaran tersebut. Dalam bola 3 dimensi, radius menghubungkan titik pusat bola dengan satu titik pada permukaan bola. Terdapat dua macam metode dalam menghitung jari-jari kelengkungan (*Radius Of Curvature)* yaitu dengan metode *Cylindrometer* dan A*rc-chord Method* [10].

2.4.1 Metode *Cylindrometer*

Metode non-optik dalam mengukur radius kelengkungan permukaan melibatkan konstruksi instrumen yang mirip dengan spherometer. Instrumen ini memiliki dua kaki tetap dan satu kaki di antara dua kaki tetap. Kaki tengah adalah kepala mikrometer yang dapat dipindah ke atas atau ke bawah dan gerakannya sudah diketahui. Titik nol instrumen adalah ketika tiga kaki menyentuh permukaan yang datar. Kemudian instrumen diletakkan di permukaan melengkung dan dicatat pada skala mikrometer yang melekat pada kaki pusat. Jarak antara dua kaki luar dapat dipilih dengan mudah (dimisalkan 20 cm untuk permukaan besar jari-jari

kelengkungan yang panjang sehingga kaki bergerak sentral berjarak 10 cm dari ujung kaki) [13]. Rumus untuk menghitung jari-jari kelengkungan ini mirip dengan yang ada pada spherometer konvensional seperti dibawah ini.



Gambar 2. 10 Diagram prinsip *cylindrometer* dasar **[13]**

Rumus dasar yang digunakan adalah sebagai berikut.

$$\sin\theta = \frac{L}{R} \text{ dan } cos\theta = \frac{(R-s)}{R} \quad (2.5)$$

Hasil pengkuadratan dan penambahan rumus diatas menghasilkan rumus dibawah ini.

$$\left(\frac{L}{R}\right)^2 + \frac{(R-s)^2}{R^2} = 1 \quad (2.6)$$

Rumus diatas kemudian disempurnakan dan dihasilkan rumus sebagai berikut.

$$L^2 + s^2 - 2Rs = 0 \quad (2.7)$$

Sehingga diperoleh rumus perhitungan jari –jari kelengkungan dari rumus diatas adalah sebagai berikut.

$$R = \frac{L^2}{2s} + \frac{s}{2} \quad (2.8)$$

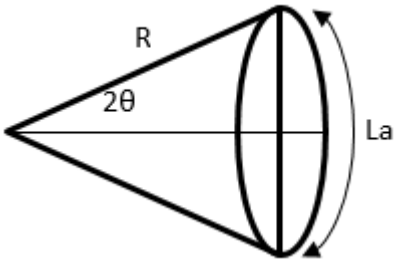
Penggunaan rumus diatas hanya berlaku pada jari-jari kelengkungan dengan bentuk lingkaran sempurna. Sehingga jika dihadapkan dengan jari-jari kelengkungan dengan lingkaran tidak sempurna atau tidak tentu, dapat diselesaikan dengan rumus dibawah ini.

$$\partial R = \frac{\partial s}{2}\left[1 + \frac{L^2}{s^2}\right] + \frac{L}{s}\partial L \quad (2.9)$$

Dengan ∂R adalah nilai ketidakpastian dalam pengukuran jari-jari kelengkungan, ∂s adalah ketidakpastian dalam pembacaan skala dan ∂L adalah ketidakpastian dalam pengukuran jarak.

2.4.2 Metode A*rc-chord*

Pada metode kedua dalam menentukan jari – jari kelengkungan memiliki beberapa berbedaan pada metode pertama. Pada metode kedua, nilai panjang kelengkungan *Chord* (La) diperhitungkan dan untuk sudut lancip segitiga sama dengan dua kali dari metode kedua [13]. Ilustrasi dari metode ini dijelaskan pada gambar 2.10 dibawah ini.



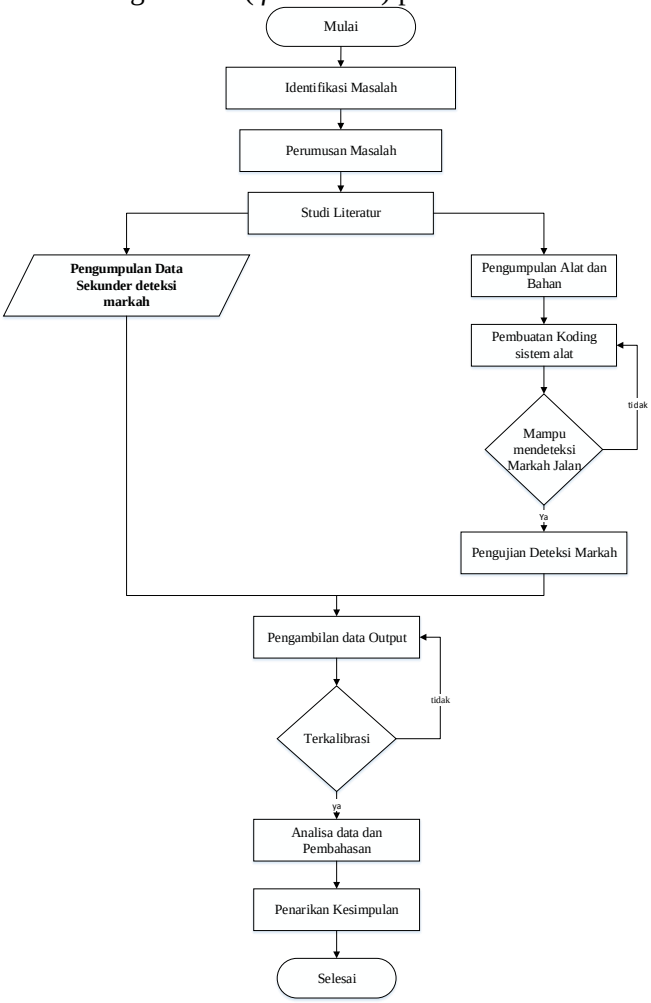
Gambar 2. 11 Diagram prinsip metode *Arc-Chord*

Rumus dalam mencari jari – jari kelengkungan pada metode kedua ini adalah sebagai berikut.

$$R = \frac{La}{2\theta} \tag{2.10}$$

BAB III
METODOLOGI PENELITIAN

Tahapan yang dilakukan dalam Tugas Akhir ini ditampilkan dengan sebuah diagram alir (*flowchart*) pada Gambar 3.1



Gambar 3. 1 Diagram alir penelitian

3.1 Identifikasi Masalah

Langkah pertama yang dilakukan pada pengerjaan tugas akhir ini adalah mengidentifikasi masalah. Pengambilan topik tugas akhir ini berawal dari hasil survey mengenai minat dan ketertarikan masyarakat dunia tentang hadirnya teknologi *Self Driving Car* yang dimana teknologi tersebut masih tampak asing di mata warga Indonesia.

3.2 Perumusan Masalah

Setelah melakukan identifikasi masalah, dilanjutkan dengan tahapan perumusan masalah. Berdasarkan identifikasi masalah yang telah dilakukan, dapat diketahui bahwa permasalahan utama yaitu di Indonesia, metode dalam menghasilkan teknologi *Self Driving Car* masih belum ada dan harga jual maupun riset tentang teknologi tersebut masih besar.

3.3 Studi Literatur

Langkah berikutnya adalah studi literatur, yaitu pada tahap ini dilakukan studi terhadap beberapa literatur dari referensi text book, manual book, ataupun jurnal-jurnal ilmiah yang memuat materi-materi berkaitan dengan penelitian yang akan dilakukan. Dari studi literatur ini, diharapkan dapat membantu peneliti dalam menentukan langkah-langkah dan pola pikir yang akan dijadikan acuan dalam menyelesaikan permasalahan pada penelitian ini. Materi-materi yang berkaitan dengan penelitian tugas akhir ini yaitu deteksi Marka jalan menggunakan sensor kamera, cara kerja *Raspberry* dan *Open Source program Python*, metode *Image Prosessing* dalam menentukan Marka jalan, penggunaan dan kelebihan dari Raspberry Pi sebagai basis pemrograman.

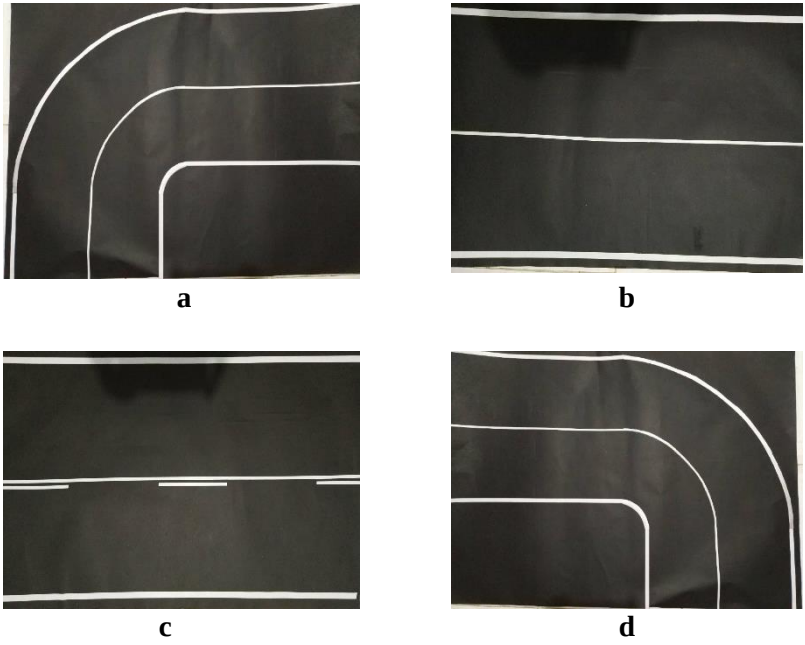
3.4 Pengumpulan Alat dan Bahan

Setelah mendapatkan literatur yang diperlukan dalam penelitian tugas akhir, langkah selanjutnya yaitu dilakukan pengumpulan alat dan bahan. Alat yang dibutukkan yaitu Raspberry Pi tipe 2, Camera Pi versi 2, *charger micro* USB, layar *monitor*, *keyboard* dan *mouse* sebagai penunjang Raspberry Pi, purwarupa mobil dengan rasio 1:16 dan marka jalan dengan rasio 1:16 dari marka yang sebenarnya dengan spesifikasi marka membujur, lurus dan melintang dengan sudut dan jari – jari

kelengkungan yang telah ditentukan. Sedangkan bahan yang diperlukan adalah sistem operasi Raspbian, Aplikasi dengan Bahasa pemrograman Python versi 3.6 dan *Application Programming Interface* berupa Open CV.

3.5 Purwarupa Marka

Langkah selanjutnya setelah dilakukan pengumpulan alat dan bahan, yaitu merancang purwarupa marka. Purwarupa yang dibuat yaitu memiliki rasio 1:16 dari marka asli. Purwarupa ini dibuat pada kertas karton ukuran A0 (84,1 x 118,9 cm) berwarna hitam dimana marka dilapisi oleh Spidol warna putih. Berikut hasil Purwarupa marka yang dibuat untuk uji deteksi.



a **b**

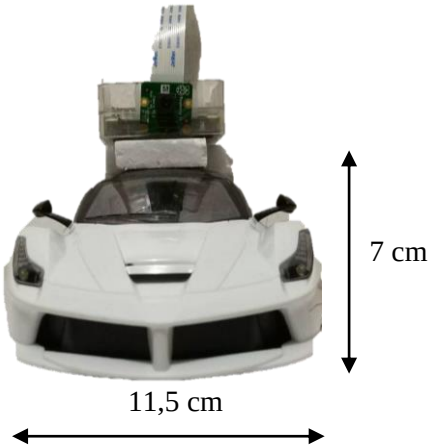
c **d**

Gambar 3.2
a) Purwarupa marka belok kanan
b) Purwarupa marka membujur utuh lurus
c) Purwarupa marka membujur lurus putus
d) Purwarupa marka belok kiri

Ukuran purwarupa yang dibuat memiliki total lebar jalan sebesar 45 cm dan total jari-jari kelengkungan sebesar 55 cm.

3.6 Purwarupa Kendaraan

Setelah dilakukan pembuatan purwarupa marka, selanjutnya dilakukan pembuatan purwarupa kendaraan. Purwarupa kendaraan yang digunakan memiliki ukuran rasio 1:16 dari aslinya dengan dimensi panjang 24 cm, lebar 11,5 cm dan tinggi 7 cm. purwarupa tersebut kemudian dipasang perangkat Raspberry Pi beserta kamera Pi yang telah terpasang di Raspberry. Berikut hasil purwarupa kendaraan untuk pendektesian.



Gambar 3.3 Purwarupa kendaraan tampak depan

Tampak depan terlihat posisi bagian depan kendaraan dengan lebar total purwarupa 11,5 cm dan tinggi purwarupa 7 cm.

sensor dan Raspberry Pi 2 diletakkan diatas kendaraan purwarupa dengan sudut kamera terhadap bidang datar sebesar 80 derajat.

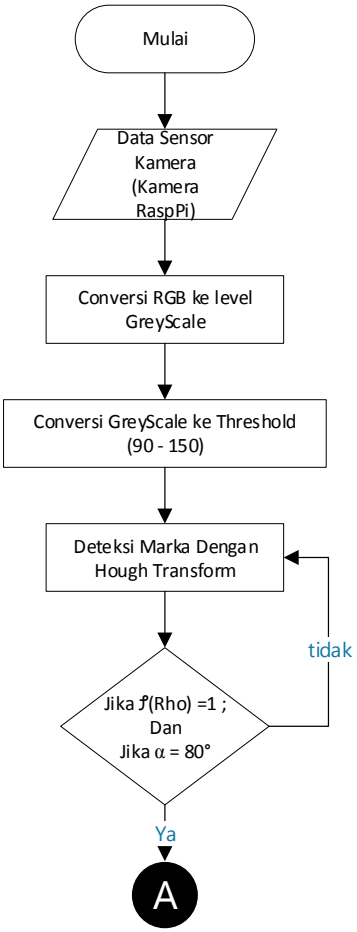
Gambar 3.4 Purwarupa Kendaraan Tampak Samping

Pada penempatan Raspberry Pi 2, diberi tumpuan berupa *sterofoam* persegi dengan ketebalan 1,2 cm. sehingga total ketinggian kendaraan dengan purwarupa sebesar 7 cm. Untuk penempatan kamera, posisi diletakkan diatas Purwarupa dengan sudut terhadap bidang datar sebesar 80º.

3.7 Pengkodingan sistem alat

Langkah selanjutnya dilanjutkan dengan pengkondingan sistem alat, dimana seluruh kodingan dibuat dalam perangkat lunak Python 3.6. pengkodingan berupa pengolahan citra gambar video yang didapat secara *real time* pada sensor kamera Rasp Pi versi 2. Hasil pengolahan citra berupa deteksi marka, posisi kendaraan terhadap marka dan deteksi kelengkungan dari jalan terhadap marka. Dibawah ini dijelaskan mengenai diagram alir pada pengkodingan proses pengolahan citra deteksi marka. Pada penjelasan dibawah ini dijabarkan mengenai diagram alur metode pendektesian pada penelitian. Diagram alur dimulai dari tahap inisiasi kamera hingga hasil akhir pendektesian menggunakan perangkat lunak *Python* versi 3.

Gambar 3.5 merupakan diagram alir prosedur yang dilakukan dalam pendektesian garis marka jalan. Penjelasan dalam setiap langkah dapat dijelaskan sebagai berikut :



Gambar 3.5 Diagram alur deteksi garis

- **Data Sensor Kamera**
  Pada tahap pertama yaitu memasukkan spesifikasi pengambilan data pada Kamera Pi, yaitu berupa besar resolusi, besar *Frame Per Second* citra yang ditangkap kamera, dan posisi awal kamera(apakah terbalik atau tidak). *Script* yang digunakan adalah sebagai berikut:

- image_size=(320, 192)
- camera = picamera.PiCamera()
- camera.resolution = image_size
- camera.framerate = 30

- **Conversi RGB ke Level *GreyScale***
  Tahab selanjutnya yaitu merubah citra yang diambil kamera yang berupa hasil berwarna menjadi hasil keabu-abuan. Hal ini digunakan untuk mempermudah pengolahan data warna. *Script* yang digunakan adalah sebagai berikut:
  - cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)

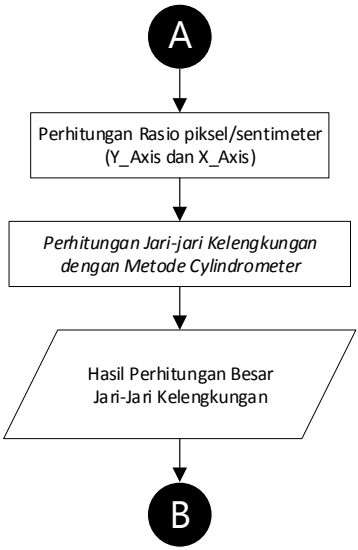
- **Conversi GreyScale ke Threshold**
  Pada tahab ini, hasil citra keabu-abuan selanjutnya diolah menjadi hitam putih untuk deteksi tepi oleh efek *Threshold.* Rentang nilai *Threshold* yaitu 90-150 dengan 2 iterasi(hitam dan putih). Metode yang digunakan adalah deteksi tepi model *Channy* yang memiliki deteksi yang sangat sempurna dengan *noise* yang paling kecil. Metode *Channy* menggabungkan deteksi tepi model Sobel dengan tambahan *Threshold. Script* yang digunakan adalah sebagai berikut:
  - self.color_thresh(image,thresh=(90, 150)
  - sobelx = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize)
  - sobely = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize)
- **Deteksi Marka dengan *Hough Transform***
  Pada tahab ini, hasil deteksi tepi kemudian diolah menggunakan metode *Hough* dalam mencari marka jalan. Dasar rumus metode diatas adalah sebagai berikut:

  $$\rho = x \cos\theta + y \sin\theta \qquad (3.1)$$

  nilai yang berubah adalah nilai pada sumbu x dan y, tergantung posisi tepi yang terdeteksi pada hasil deteksi tepi. Nilai ρ yang dicari adalah 1. Sudut menggunakan 80 derajat dari referensi kamera. Jika nilai ρ mendekati atau sama dengan 1, maka *point* terdeteksi. Setiap sekuens dicari nilai *point*, dan jika kedua *point* dihubungkan, akan membentuk sebuah garis. Akan tetapi jika *point* tidak ditemukan, maka

program secara otomatis mencari *point* tersebut dari referensi deteksi tepi *Channy* sampai menemukan *point* dari setiap citra yang ditangkap kamera Pi. Langkah selanjutnya yaitu dijelaskan mengenai deteksi parameter penunjang pertama yaitu deteksi jari-jari kelengkungan pada belokan marka. *Script* yang digunakan adalah sebagai berikut:

- lines = cv2.HoughLinesP(mag_binary, rho=1, theta=np.pi/180, threshold=30, minLineLength=40, maxLineGap=100)



Gambar 3.6 Diagram alur deteksi jari-jari kelengkungan

Gambar 3.6 merupakan prosedur yang dilakukan dalam pendektesian jari-jari kelengkungan marka. Penjelasan dalam setiap langkah dapat dijelaskan sebagai berikut :

- **Perhitungan Rasio Piksel per Sentimeter**
  Sebelum pada tahab pendektesian jari-jari kelengkungan, sebelumnya dicari nilai besar dari piksel per sentimeter dari citra yang diambil. Hal ini dilakukan untuk memberi

keakuratan besar pengukuran pada citra dua dimensi terhadap ukuran sebenarnya. Terdapat dua sumbu rasio yang dicari yaitu sumbu x dan sumbu y. *Script* yang digunakan adalah sebagai berikut:

- ym_per_pix = 0.2 # meter per pixel pada y dimensi
- xm_per_pix = 0.01 # meter per pixel pada x dimensi # pembuatan persamaan x,y pada 2D
- left_fit_cr = np.polyfit(self.bestyl*ym_per_pix, self.bestxl*xm_per_pix, 2)
- right_fit_cr = np.polyfit(self.bestyr*ym_per_pix, self.bestxr*xm_per_pix, 2)

- **Perhitungan Jari-jari Kelengkungan**

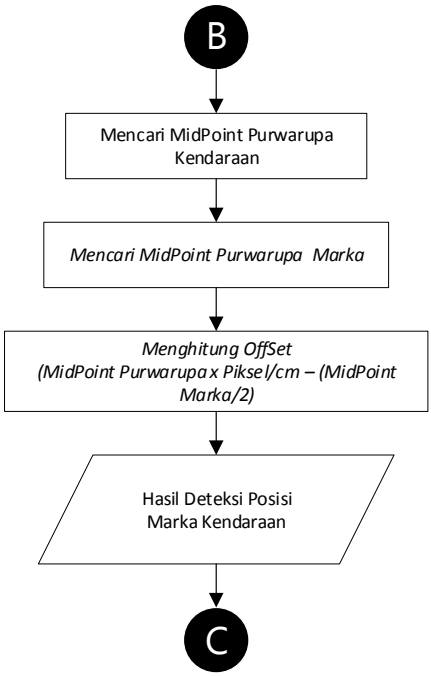
  Pada tahab selanjutnya yaitu mendeteksi besar jari-jari kelengkungan pada belokan marka menggunakan metode *Cylindrometer* pada kelengkungan tidak sempurna. Rumus yang digunakan adalah sebagai berikut:

$$\partial R = \frac{\partial s}{2}\left[1 + \frac{L^2}{s^2}\right] + \frac{L}{s}\partial L \qquad (3.2)$$

  Script yang digunakan adalah sebagai berikut:

- left_curverad
  = ((1 + (2*left_fit_cr[0]*y_eval*ym_per_pix + left_fit_cr[1])**2)**1.5) / np.absolute(2*left_fit_cr[0])
- right_curverad
  =((1+(2*right_fit_cr[0]*y_eval*ym_per_pix + right_fit_cr[1])**2)**1.5) / np.absolute(2*right_fit_cr[0])

  Setelah didapatkan nilai jari-jari kelengkungan, langkah selanjutnya adalah pendektesian parameter posisi kendaraan terhadap marka yang dijelaskan pada diagram alir dibawah ini.



Gambar 3.7 Diagram alir deteksi posisi kendaraan

Gambar 3.7 merupakan prosedur yang dilakukan dalam pendektesian posisi kendaraan terhadap marka. Penjelasan dalam setiap langkah dapat dijelaskan sebagai berikut :

- **Mencari *MidPoint* Purwarupa Kendaraan**
  Pada tahab selanjutnya yaitu dicari nilai titik tengah pada purwarupa kendaraan. Nilai tersebut digunakan sebagai acuan kendaraan pada marka jalan. *Script* yang digunakan adalah sebagai berikut:
  - midpoint_car = self.im_shape[1]/2.0
- **Mencari *MidPoint* Purwarupa Marka**
  Pada tahab selanjutnya yaitu dicari niai titik tengah pada purwarupa marka. Nilai tersebur digunakan sebagai acuan

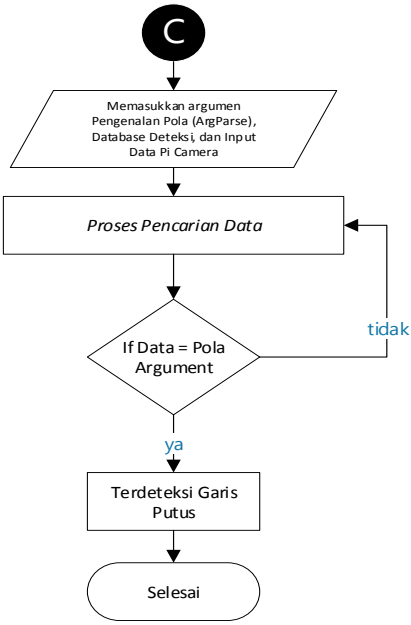
menentukan nilai *Offset* Kendaraan pada marka. *Script* yang digunakan adalah sebagai berikut:

- midpoint_lane
  =(right_fit_cr[0]*(y_eval**2) + right_fit_cr[1]*y_eval + right_fit_cr[2]) + \(left_fit_cr[0]*(y_eval**2) + left_fit_cr[1]*y_eval + left_fit_cr[2])

- **Mencari Nilai *Offset***

  Pada tahab ini dicari nilai *offset* purwarupa pada marka jalan. *Offset* dicari dengan nilai tengah purwarupa dikurangi dengan nilai tengah marka dibagi dengan 2. *Script* yang digunakan adalah sebagai berikut:

  - offset = midpoint_car*xm_per_pix - midpoint_lane/2



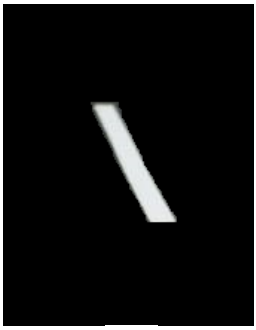
Gambar 3.8 Diagram alir deteksi garis putus

Gambar 3.8 merupakan prosedur yang dilakukan dalam pendektesian garisputus pada marka. Penjelasan dalam setiap langkah dapat dijelaskan sebagai berikut :

- **Memasukkan Argumen Pengenalan Pola**
  Pada tahab berikut, dimasukkan *database* mengenai garis putus berupa gambar. Gambar tersebut merupakan gambar garis putus hasil pengambilan citra gambar yang telah diseleksi. Hasil seleksi terletak pada gambar 3.9 dibawah ini.

a

b



c

Gambar3.9 a) Gambar garis mentah
b) Hasil *Crop* marka putus
c) Hasil *Trheshold* Marka putus

Script yang digunakan adalah sebagai berikut:

- ap = argparse.ArgumentParser()
  ap.add_argument("-i", "--image", help="../c/mark

```
true4.png")
args = vars(ap.parse_args())
```

- **Proses Pencarian Data**

  Pada tahab ini setelah dilakukan pemasukan data marka kedalam *database*. Pencocokan dilakukan dengan diambil data citra pada sensor kemudian dicocokkan dengan gambar garis putus pada *database.* Proses pencarian dilakukan secara terus menerus hingga diperoleh hasil yang cocok.

3.8 Pengujian Deteksi Marka

Setelah dilakukan pengkodingan, langkah selanjutnya adalah dilakukan pengujian alat purwarupa dalam pendektesian marka. Dalam tahap pengujian dilakukan penyempurnaan dari kodingan pada perangkat lunak Python jika ditemukan suatu *error Script* dan gagalnya pendektesian.

3.9 Pengambilan Data Output

Setelah program berhasil mendekteksi, langkah selanjutnya yaitu pengambilan data pada marka dengan spesifikasi purwarupa marka yaitu dengan rasio 1:16 dengan marka asli. Pada tahap ini dilakukan nilai kalibrasi terhadap besaran Pixel pada tiap satuan sentimeter untuk dihasilkan nilai pendektesian yang terbaik.

3.10 Analisa Data dan Pembahasan

Setelah melakukan pengambilan data, selanjutnya dilakukan analisa data hasil pendektesian dari purwarupa dan pembahasan. Data yang diambil berupa hasil deteksi marka, parameter penunjang seperti jari-jari kelengkungan, posisi kendaraan pada posisi marka, dan status kendaraan teradap garis tengah marka. Hasil parameter tersebut kemudian dicari nilai koreksi / *error* pada pengambilan data dengan perbedaan dua kecepatan.

3.11 Penarikan kesimpulan

Pada tahap terakir dilakukan penarikan kesimpulan percobaan.

Halaman Ini Sengaja Dikosongkan

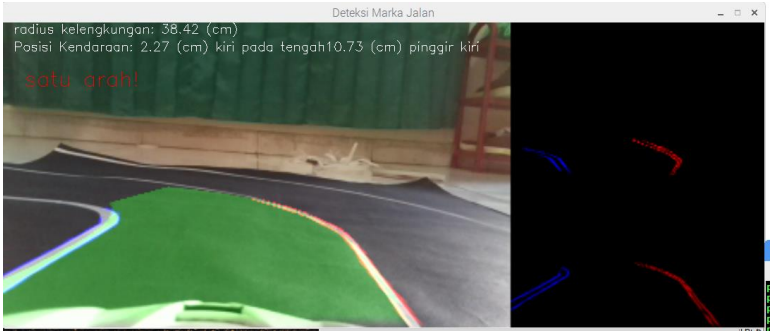
BAB IV
ANALISIS DATA DAN PEMBAHASAN

4.1 Hasil Deteksi Marka

Hasil deteksi marka dilakukan dengan cara mendapatkan nilai parameter hasil pengolahan citra yaitu parameter berupa deteksi marka, posisi kendaraan terhadap marka dan deteksi kelengkungan dari jalan terhadap marka pada marka melintang jalan lurus, membujur pada jalan lurus dan tikungan, dan serong pada jalan lurus. Parameter didapatkan dengan melakukan *screen shot* pada program saat mobil purwarupa berjalan pada kecepatan tetap dan berada pada titik yang telah ditandai. Data *Screen shot* yang diambil pada setiap marka berjumlah 10 dan diambil 5 data terbaik.

4.1.1 Hasil Deteksi Marka berbelok

Berikut ini pada gambar 4.1 ditampilkan hasil pembacaan Raspberry Pi pada marka membujur pada jalan berbelok kiri. Ukuran sebenarnya pada purwarupa marka adalah memiliki jari-jari kelengkungan 55 cm, posisi tengah marka terhadap kendaraan 2,5 cm, dan posisi tepi marka terhadap kendaraan 9 cm. Nilai error didapatkan dari nilai sebenarnya dikurangi dengan nilai hasil pembacaan dibagi dengan nilai sebenarnya.



Gambar 4.1 Hasil deteksi marka belok kiri

Pada tabel 4.1 ditampilkan hasil pembacaan Raspberry Pi pada marka membujur pada jalan berbelok kiri. Parameter yang terdapat pada tabel 4.1 berupa nilai jari-jari kelengkungan, posisi kendaraan terhadap marka tengah, posisi kendaraan terhadap marka tepi dan nilai error hasil pembacaan pada setiap percobaan.

Tabel 4.1 Hasil Deteksi Marka Membujur belok kiri

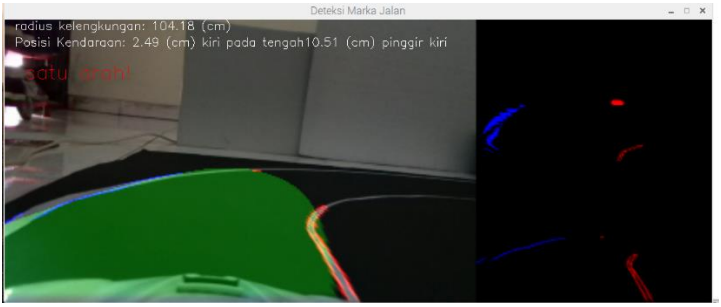
Percobaan Ke	Kecepatan (cm/s)	Jari-jari Kelengkungan(cm)	Posisi tengah (cm)	Posisi tepi (cm)	Error (%)
1	74,4	41,3	2,3	10,7	12,94949
2		43,7	2,3	10,7	11,85859
3		54,1	2,3	10,7	7,131313
4		41,1	2,3	10,7	13,0404
5		38,5	2,3	10,7	14,22222
6		38,2	2,3	10,7	14,35859
7		71,7	2,3	10,7	14,31313
8		74,6	2,3	10,7	15,63131
9		40,28	2,2	10,7	14,41313
10		56,5	2,3	10,7	7,40404
11		51,7	2,3	10,7	8,222222
12		51,1	2,3	10,7	8,494949
13		76,4	2,3	10,7	16,44949
14		55,2	2,3	10,7	6,813131
15		56,7	2,3	10,7	7,494949
16		52,13	2,2	10,8	9,304545
17		59	2,2	10,8	9,818182
18		58,3	2,2	10,8	9,5
19		48,7	2,3	10,7	9,585859
20		33,1	2,3	10,7	16,67677
21	115,2	139,5	2,3	10,7	45,13131
22		98,5	2,3	10,7	26,49495
23		83,2	2,3	10,7	19,5404
24		84,7	2,3	10,7	20,22222
25		86,4	2,3	10,7	20,99495
26		101,3	2,3	10,7	27,76768
27		78,6	2,3	10,7	17,44949
28		131,5	2,3	10,7	41,49495
29		127,9	2,3	10,7	39,85859
30		148,3	2,3	10,7	49,13131
31		113,2	2,3	10,7	33,17677
32		77,8	2,2	10,8	18,36364
33		84,3	2,3	10,7	20,0404
34		86,1	2,3	10,7	20,85859
35		84,6	2,3	10,7	20,17677
36		84,9	2,3	10,7	20,31313

Tabel 4.1 Lanjutan

Percobaan Ke	Kecepatan (cm/s)	Jari-jari Kelengkungan(cm)	Posisi tengah (cm)	Posisi tepi (cm)	Error (%)
37	115,2	84,8	2,2	10,7	21,26768
38		86,1	2,3	10,7	20,85859
39		84,3	2,2	10,8	21,31818
40		84,6	2,3	10,7	20,17677
			Error Kecepatan 1		11,38412
			Error Kecepatan 2		26,23182
			Total Error		18,80797

Pada hasil percobaan pada deteksi marka membujur belok kiri ditemukan rata-rata nilai error pada variasi kecepatan pertama 74,4 cm/s yaitu sebesar 11,38% dan rata-rata error pada variasi kecepatan kedua 115,2 cm/s yaitu sebesar 26,23%. Total error yang diperoleh pada deteksi marka belok kanan yaitu sebesar 18,81%.

Berikut ini pada gambar 4.2 ditampilkan hasil pembacaan Raspberry Pi pada marka membujur pada jalan berbelok kanan. Ukuran sebenarnya pada purwarupa marka adalah memiliki jari-jari kelengkungan 55 cm, posisi tengah marka terhadap kendaraan 2,5 cm, dan posisi tepi marka terhadap kendaraan 9 cm. Nilai error didapatkan dari nilai sebenarnya dikurangi dengan nilai hasil pembacaan dibagi dengan nilai sebenarnya.



Gambar 4.2 Hasil deteksi marka belok kanan

Pada tabel 4.2 ditampilkan hasil pembacaan Raspberry Pi pada marka membujur pada jalan berbelok kanan. Parameter yang terdapat pada tabel 4.2 berupa nilai jari-jari kelengkungan, posisi kendaraan terhadap marka tengah, posisi kendaraan terhadap marka tepi dan nilai error hasil pembacaan pada setiap percobaan.

Tabel 4.2 Hasil Marka Membujur Belok Kanan

Percobaan Ke	Kecepatan (cm/s)	Jari-jari Kelengkungan(cm)	Posisi tengah (cm)	Posisi tepi (cm)	Error (%)
1	74,4	57,3	2	10,9	11,32323
2		45,9	2,3	10,7	10,85859
3		75,8	2,3	10,7	16,17677
4		44,2	2,3	10,7	11,63131
5		45,3	2,3	10,7	11,13131
6		71,7	2,2	10,8	15,59091
7		66,9	2,2	10,8	13,40909
8		72,9	2,2	10,8	16,13636
9		68,9	2,4	10,5	11,48485
10		70,8	2,2	10,8	15,18182
11		66,2	2,3	10,7	11,81313
12		63,7	2,3	10,7	10,67677
13		72,6	2,3	10,7	14,72222
14		78,7	2,2	10,8	18,77273
15		66,3	2,3	10,7	11,85859
16		74,7	2,2	10,8	16,95455
17		68,6	2,2	10,8	14,18182
18		74,3	2,2	10,8	16,77273
19		66,5	2,3	10,7	11,94949
20		59,3	2,3	10,6	8,39899
21	115,2	95,8	2,2	10,7	26,26768
22		84,6	2,2	10,8	21,45455
23		80,1	2,3	10,7	18,13131
24		95,9	2,3	10,7	25,31313
25		90,6	2,3	10,6	22,62626
26		98,2	2,3	10,7	26,35859
27		84,4	2,2	10,7	21,08586
28		97	2,3	10,7	25,81313
29		91,6	2,3	10,7	23,35859
30		91,9	2,2	10,8	24,77273
31		83,7	2,2	10,8	21,04545
32		80,4	2,2	10,8	19,54545
33		93,3	2,2	10,8	25,40909
34		98,4	2,3	10,7	26,44949

Tabel 4.2 Lanjutan

Percobaan Ke	Kecepatan (cm/s)	Jari-jari Kelengkungan(cm)	Posisi tengah (cm)	Posisi tepi (cm)	Error (%)
35	115,2	90	2,3	10,7	22,63131
36		111,3	2,3	10,7	32,31313
37		83,2	2,2	10,7	20,5404
38		83,4	2,3	10,7	19,63131
39		80,54	2,2	10,8	19,60909
40		90,2	2,3	10,7	22,72222
			Error Kecepatan 1		13,45126
			Error Kecepatan 2		23,25394
			Total Error		18,3526

Pada hasil percobaan pada deteksi marka membujur belok kanan ditemukan rata-rata nilai error pada variasi kecepatan pertama 74,4 cm/s yaitu sebesar 13,45% dan error pada variasi kecepatan kedua 115,2 cm/s yaitu sebesar 23,3%. Total error yang diperoleh pada deteksi marka belok kiri yaitu sebesar 18,35%.

4.1.2 Hasil Deteksi Marka Lurus

Berikut ini pada gambar 4.3 ditampilkan hasil pembacaan Raspberry Pi pada marka membujur lurus putus. Ukuran sebenarnya pada purwarupa marka adalah memiliki jari-jari kelengkungan tak hingga, posisi tengah marka terhadap kendaraan 2,5 cm, dan posisi tepi marka terhadap kendaraan 9 cm. Nilai error didapatkan dari nilai sebenarnya dikurangi dengan nilai hasil pembacaan dibagi dengan nilai sebenarnya.

Gambar 4.3 Hasil deteksi marka lurus putus

Pada tabel 4.3 ditampilkan hasil pembacaan Raspberry Pi pada marka membujur pada jalan lurus putus. Parameter yang terdapat pada tabel 4.3 berupa nilai jari-jari kelengkungan, posisi kendaraan terhadap marka tengah, posisi kendaraan terhadap marka tepi dan nilai error hasil pembacaan pada setiap percobaan.

Tabel 4.3 Hasil Deteksi Marka Membujur Lurus Putus

Percobaan Ke	Kecepatan (cm/s)	Jari-jari Kelengkungan(cm)	Posisi tengah (cm)	Posisi tepi (cm)	Error (%)
1	74,4	∞	2,3	10,7	6,722222
2		∞	2,3	10,5	6,166667
3		∞	2,2	10,7	7,722222
4		∞	2,3	10,7	6,722222
5		∞	2,3	10,7	6,722222
6		∞	2,3	10,5	6,166667
7		∞	2,4	10,6	5,444444
8		∞	2,3	10,6	6,444444
9		∞	2,3	10,6	6,444444
10		∞	2,4	10,6	5,444444
11		∞	2,3	10,7	6,722222
12		∞	2,3	10,5	6,166667
13		∞	2,2	10,7	7,722222
14		∞	2,3	10,7	6,722222
15		∞	2,3	10,7	6,722222
16		∞	2,3	10,5	6,166667

Tabel 4.3 Lanjutan

Percobaan Ke	Kecepatan (cm/s)	Jari-jari Kelengkungan(cm)	Posisi tengah (cm)	Posisi tepi (cm)	Error (%)
17	74,4	∞	2,4	10,6	5,444444
18		∞	2,3	10,6	6,444444
19		∞	2,3	10,6	6,444444
20		∞	2,4	10,6	5,444444
21	115,2	∞	2,3	10,7	6,722222
22		∞	2,3	10,7	6,722222
23		∞	2,3	10,7	6,722222
24		∞	2,3	10,7	6,722222
25		∞	2,3	10,7	6,722222
26		∞	2,3	10,7	6,722222
27		∞	2,3	10,7	6,722222
28		∞	2,3	10,7	6,722222
29		∞	2,3	10,7	6,722222
30		∞	2,3	10,7	6,722222
31		∞	2,3	10,7	6,722222
32		∞	2,3	10,7	6,722222
33		∞	2,3	10,7	6,722222
34		∞	2,3	10,7	6,722222
35		∞	2,3	10,7	6,722222
36		∞	2,3	10,7	6,722222
37		∞	2,3	10,7	6,722222
38		∞	2,3	10,7	6,722222
39		∞	2,3	10,7	6,722222
40		∞	2,3	10,7	6,722222
			Error Kecepatan 1		6,4
			Error Kecepatan 2		6,722222
			Total Error		6,561111

Pada hasil percobaan pada deteksi marka membujur lurus putus ditemukan rata-rata nilai error pada variasi kecepatan pertama 74,4 cm/s yaitu sebesar 6,4% dan error pada variasi kecepatan kedua 115,2 cm/s yaitu sebesar 6,72%. Total error yang diperoleh pada deteksi marka lurus putus yaitu sebesar 6,56%.

Berikut ini pada gambar 4.4 ditampilkan hasil pembacaan Raspberry Pi pada marka membujur lurus utuh. Ukuran sebenarnya

pada purwarupa marka adalah memiliki jari-jari kelengkungan tak hingga, posisi tengah marka terhadap kendaraan 2,5 cm, dan posisi tepi marka terhadap kendaraan 9 cm. Nilai error didapatkan dari nilai sebenarnya dikurangi dengan nilai hasil pembacaan dibagi dengan nilai sebenarnya.

Gambar 4.4 Hasil deteksi marka lurus utuh

Pada tabel 4.4 ditampilkan hasil pembacaan Raspberry Pi pada marka membujur pada jalan lurus utuh. Parameter yang terdapat pada tabel 4.4 berupa nilai jari-jari kelengkungan, posisi kendaraan terhadap marka tengah, posisi kendaraan terhadap marka tepi dan nilai error hasil pembacaan pada setiap percobaan.

Tabel 4.4 Hasil Deteksi Marka Membujur Lurus Utuh

Percobaan Ke	Kecepatan (cm/s)	Jari-jari Kelengkungan(cm)	Posisi tengah (cm)	Posisi tepi (cm)	Error (%)
1	74,4	∞	2,3	10,7	6,722222
2		∞	2,3	10,5	6,166667
3		∞	2,2	10,6	6,444444
4		∞	2,3	10,7	7,722222
5		∞	2,3	10,7	6,722222
6		∞	2,3	10,8	8
7		∞	2,4	10,6	6,444444
8		∞	2,3	10,6	6,444444
9		∞	2,3	10,6	6,444444
10		∞	2,4	10,8	8

Tabel 4.4 Lanjutan

Percobaan Ke		Jari-jari Kelengkungan(cm)	Posisi tengah (cm)	Posisi tepi (cm)	Error (%)
11		∞	2,3	10,6	6,444444
12		∞	2,3	10,7	7,722222
13		∞	2,2	10,7	5,722222
14		∞	2,3	10,6	5,444444
15		∞	2,3	10,7	6,722222
16		∞	2,3	10,5	6,166667
17		∞	2,4	10,6	5,444444
18		∞	2,3	10,6	6,444444
19		∞	2,3	10,7	5,722222
20		∞	2,4	10,6	5,444444
21	115,2	∞	2,3	10,7	6,722222
22		∞	2,3	10,7	6,722222
23		∞	2,3	10,7	6,722222
24		∞	2,3	10,7	6,722222
25		∞	2,3	10,7	6,722222
26		∞	2,3	10,7	7,722222
27		∞	2,3	10,7	6,722222
28		∞	2,3	10,8	8
29		∞	2,3	10,7	6,722222
30		∞	2,3	10,7	6,722222
31		∞	2,3	10,7	6,722222
32		∞	2,3	10,7	6,722222
33		∞	2,3	10,7	6,722222
34		∞	2,3	10,7	6,722222
35		∞	2,3	10,7	7,722222
36		∞	2,3	10,7	6,722222
37		∞	2,3	10,8	8
38		∞	2,3	10,7	6,722222
39		∞	2,3	10,7	6,722222
40		∞	2,3	10,7	6,722222
			Error Kecepatan 1		6,519444
			Error Kecepatan 2		6,95
			Total Error		6,734722

Pada hasil percobaan pada deteksi marka membujur lurus utuh ditemukan rata-rata nilai error pada variasi kecepatan pertama 74,4 cm/s yaitu sebesar 6,52% dan error pada variasi kecepatan

kedua 115,2 cm/s yaitu sebesar 6.9%. Total error yang diperoleh pada deteksi marka lurus utuh yaitu sebesar 6,73%.

4.2 Pembahasan

Dari hasil percobaan yang telah dilakukan, pada pendektesian keseluruhan pada gambar 4.1, 4.2, 4.3, 4.4 marka memiliki tingkat keakuratan 100% yaitu dapat mendeteksi posisi marka yang benar. Nilai tersebut diperoleh dengan memperhatikan sudut *hough* dan *value* pada *Threshold* citra gambar. Dengan nilai *threshold* pada 90 – 150, dan sudut *Hough* sebesar 80º, marka dapat dideteksi dengan baik. Untuk hasil parameter tambahan berupa jari – jari kelengkungan dari tikungan, posisi kendaraan terhadap garis tengah marka, posisi kendaraan terhadap garis kiri dan kanan marka, dan perintah mengenai perubahan jalur marka memiliki beberapa tingkat *error* yang terlihat dari jarak sebenarnya. Dari parameter diatas, diuji dalam dua kecepatan berbeda yaitu 74,4 cm/s dan 115,2 cm/s. Pada percobaan yang tercantum di tabel 4.1 untuk marka belok ke kiri pada kecepatan 74,4 cm/s memiliki total rata-rata *error* yaitu 11,38% dan untuk kecepatan 115,2 cm/s memiliki total rata-rata *error* yaitu 26,23%. Nilai *error* pada kecepatan 115,2 cm/s memiliki tingkat paling tinggi dibanding dengan kecepatan 74,4 cm/s dikarenakan tingkat *refresh rate* kamera dan *Frame Per Second* pada Rasp kamera adalah 15 FPS. Sehingga dimungkinkan pada kecepatan terlalu tinggi, tingkat deteksi pada marka dan nilai jari – jari kelengkungan/*Radius Of Curvature* memiliki tingkat keakuratan yang sedikit. Untuk menaikkan keakuratan, diperlukan peningkatan performa Update *refresh rate* deteksi marka dan peningkatan nilai Frame Per Second dari kamera Pi. Hal serupa juga terjadi pada deteksi marka belok kekanan yang tercantum di tabel 4.2 dimana rata-rata nilai *error* pada kecepatan 115,2 cm/s sangat tinggi yaitu 23,25% dibanding dengan kecepatan 74,4 cm/s sebesar 13,45%. Sedangkan untuk deteksi marka membujur lurus putus yang tercantum dalam tabel

4.3 memiliki rata-rata *error* pada kecepatan 74,4 cm/s yaitu sebesar 6,4% dan 115,2 cm/s sebesar 6,72%. hal tersebut terjadi dikarenakan marka yang dideteksi adalah marka lurus, sehingga dalam pendektesian jari – jari kelengkungan tidak perlu dilakukan karena bernilai tak hingga. Selanjutnya untuk deteksi marka membujur lurus utuh yang tercantum pada tabel 4.4 memiliki rata-rata *error* pada kecepatan 74,4 cm/s yaitu sebesar 6,52% dan 115,2 cm/s sebesar 6,95%. hal tersebut juga terjadi dikarenakan marka yang dideteksi adalah marka lurus, sehingga dalam pendektesian jari – jari kelengkungan tidak perlu dilakukan karena bernilai tak hingga. Pada penelitian yang dibuat ini jika dibandingkan dengan penelitian yang telah dilakukan yaitu oleh Marcos Paulo Batista tentang "*Lane Detection and Estimation using Perspective Image*" [14] memiliki beberapa persamaan dan perbedaan. Persamaan tersebut yaitu sama-sama menggunakan metode *Hough Transform* dalam mendeteksi marka jalan dengan menggunakan perspektif dari marka. Akan tetapi dalam mendeteksi posisi kendaraan, jari-jari kelengkungan dan keakuratan lebih dalam mendeteksi marka tengah pada penelitian Marcos tidak diterapkan.

Halaman Ini Sengaja Dikosongkan

BAB V
PENUTUP

5.1 Kesimpulan

Adapun kesimpulan yang didapatkan dari Tugas Akhir ini yakni sebagai berikut :

- Telah dapat dirancang dan dibangun sistem deteksi marka jalan dengan Metode *Hough Transform* berbasis Raspberry Pi.
- Sistem deteksi marka dapat mendeteksi dengan baik pada kecepatan pertama yaitu 74,4 cm/s dengan total rata-rata *error* pada marka belok kiri 11,38%, marka belok kanan 13,45%, marka lurus putus 6,4% dan lurus utuh 6,51%.
- Sistem deteksi marka kurang dapat mendeteksi dengan baik pada kecepatan kedua yaitu 115,2 cm/s dengan rata-rata *error* pada marka belok kiri 26,23%, marka belok kanan 23,25%. Sedangkan untuk marka lurus memiliki deteksi sangat baik pada kecepatan kedua karena untuk perhitungan kelengkungan marka tidak diperhitungkan. Nilai rata-rata marka lurus putus pada kecepatan kedua adalah 6,72% dan lurus utuh 6,95%.
- Variasi kecepatan dan spesifikasi control yang digunakan yaitu Raspberry Pi sangat berpengaruh pada pembaruan pendektesian marka terutama pada deteksi jari –jari kelengkungan.

5.2 Saran

Adapun saran yang dapat diberikan untuk penelitian selanjutnya antara lain :

- Dibutuhkan peningkatan *Frame Per Second* pada kamera Pi dan dilakukan kalibrasi kembali terhadap nilai Pixel pada satuan sentimeter sehingga dapat ditemukan nilai pendektesian yang lebih mendekati pada nilai yang sesungguhnya.
- Pada variasi kecepatan, menggunakan potensio meter dalam pengubahan voltase agar nilai kecepatan yang terukur memiliki nilai yang baik.

Halaman Ini Sengaja Dikosongkan

DAFTAR PUSTAKA

[1] N. McCarthy, "Statista, The Statistic Portal," 2 desember 2015. [Online]. Available: https://www.statista.com/chart/4092/how-do-consumers-feel-about-self-driving-cars/. [Diakses 15 Maret 2018].

[2] N. Wilson, "spesikamu.com," 31 januari 2017. [Online]. Available: https://spasikamu.com/waymo-luncurkan-teknologi-self-driving-car-terbaru-nw/. [Diakses 10 maret 2018].

[3] R. Pi, "OpenSource Raspberry Pi," [Online]. Available: https://opensource.com/resources/raspberry-pi.

[4] J. Brodkin, "arstechnica.com," Ars Technica, februari 2015. [Online]. Available: https://arstechnica.com/information-technology/2015/02/raspberry-pi-2-arrives-with-quad-core-cpu-1gb-ram-same-35-price/.

[5] pccontrol, "Pengetahuan Dasar dan Pemrograman Raspberry Pi," [Online]. Available: https://pccontrol.wordpress.com/2014/06/17/pengetahuan-dasar-dan-pemrograman-raspberry-pi/.

[6] Roderta, Rancang Bangun Sistem Peringatan Keamanan Pada Mobil Berdasarkan Garis Markah Jalan Menggunakan Kamera, Surabaya, 2014.

[7] C. R. J. a. C. R. Kelber, "A lane departure warning system based on a linear-parabolic lane model," *IEEE,* 2004.

[8] J. Kepar, "Data Parse," 12 Februari 2014. [Online]. Available: https://codesaya.com/diskusi/d/mem-parsing-data-0814931274059/.

[9] Y. B. L. Hao Wu, "A Lane Detection Approach for Driver Assistance," *IEEE,* desember 2009.

[10] S. M.-S. A. M. Mirko Meutery, "A Novel Approach to Lane Detection and Tracking," *IEEE,* 7 oktober 2009.

[11] A. W. E. W. R. Fisher, "Hough Transform," HIPR2, 2003.

[Online]. Available: https://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm.

[12] B. H. d. KSLN, "PERATURAN MENTERI PERHUBUNGAN NOMOR PM 34 TAHUN 2014," 4 September 2014. [Online]. Available: http://jdih.dephub.go.id/assets/uudocs/permen/2014/pm_34_tahun_2014.pdf.

[13] D. U. R.P. Shukla, "Measurment of Radius of Curvature of Cylindrical Surfaces," *Journal of Optics,* vol. 30, pp. 131-142, 2001.

[14] P. Y. S. D. F. W. a. D. G. Marcos Paulo Batista, "Lane Detection and Estimation using Perspective," *SBR-LARS Robotics Symposium and Robocontrol,* 2014.

LAMPIRAN A
Script Python Deteksi Marka Jalan

```
import picamera
from picamera.array import PiRGBArray
import numpy as np
import cv2
import time
import warnings
import uuid
import argparse
warnings.filterwarnings('error')

#Inisiasi Kamera yang digunakan
image_size=(320, 192)
camera = picamera.PiCamera()
camera.resolution = image_size
camera.framerate = 30
camera.vflip = True
camera.hflip = True
rawCapture = PiRGBArray(camera, size=image_size)

# proses warming up kamera pi
time.sleep(0.1)

# inisiasi argumen pengenalan objek pada marka putus
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", help="../c/mark
true4.png")
args = vars(ap.parse_args())

# Pembagian kelas deteksi marka
class Lines():
    def __init__(self):
        # ketika marka yang terdeteksi kurang lebih satu
marka
```

```
        self.detected_first = False
        # ketika marka yang terdeteksi pada iterasi terakir
        self.detected = False
        # rata - rata nilai sumbu x pada line
        self.bestxl = None
        self.bestyl = None
        self.bestxr = None
        self.bestyr = None
        # koefisien rata - rata polynomial pada iterasi terakir
        self.best_fit_l = None
        self.best_fit_r = None
        #koefisien polynomial  untuk semua hasil fit
        self.current_fit_l = None
        self.current_fit_r = None
        # radius kelengkungan suatu garis pada besaran meter
        self.left_curverad = None
        self.right_curverad = None
        #jarak kendaraan pada satuan meter pada center/pusat
dari marka
        self.offset = None
        # nilai x pada marka terdeteksi
        self.allxl = None
        self.allxr = None
        # nilai y pada marka terdeteksi
        self.allyl = None
        self.allyr = None
        # parameter kalibrasi kamera
        self.cam_mtx = None
        self.cam_dst = None
        # parameter distorsi kamera
        self.M = None
        self.Minv = None
        # bentuk image
        self.im_shape = (None,None)
        # jarak pada posisi depan dengan satuan sentimeter
        self.look_ahead = 10
        self.remove_pixels = 90
```

```
        # perbesar keluaran image
        self.enlarge = 2.5

        # peringatan pada library codingan numpy polyfit
        self.poly_warning = False

    # penerapan nilai parameter kalibrasi kamera
    def set_cam_calib_param(self, mtx, dst):
        self.cam_mtx = mtx
        self.cam_dst = dst

    # undistorsi image
    def undistort(self, img):
        return cv2.undistort(img, self.cam_mtx,
self.cam_dst, None,self.cam_mtx)

    # perolehan binary image berdasarkan warna
thresholding
    def color_thresh(self, img, thresh=(0, 255)):
        # settingan warna HSV
        hsv= cv2.cvtColor(img,
cv2.COLOR_RGB2HLS).astype(np.float)
        s_channel = hsv[:,:,2]

        # threshold channel
        s_binary = np.zeros_like(s_channel)
        s_binary[(s_channel >= thresh[0]) & (s_channel <=
thresh[1])] = 1
        return s_binary

    # perolehan binary image berdasarkan thresholding
gradien sobel
    def abs_sobel_thresh(self, sobel, thresh=(0, 255)):

        abs_sobel = np.absolute(sobel)

        max_s = np.max(abs_sobel)
```

```
        if max_s == 0:
            max_s=1

        scaled_sobel = np.uint8(255*abs_sobel/max_s)

        sbinary = np.zeros_like(scaled_sobel)
        sbinary[(scaled_sobel >= thresh[0]) & (scaled_sobel
<= thresh[1])] = 1

        return sbinary

    # perolehan binary image berdasarkan besaran gradien
thresholding sobel
    def mag_thresh(self, sobelx, sobely, mag_thresh=(0,
255)):

        abs_sobel = np.sqrt(sobelx**2 + sobely**2)

        max_s = np.max(abs_sobel)
        if max_s == 0:
            max_s=1

        scaled_sobel = np.uint8(255*abs_sobel/max_s)

        sbinary = np.zeros_like(scaled_sobel)
        sbinary[(scaled_sobel >= mag_thresh[0]) &
(scaled_sobel <= mag_thresh[1])] = 1

        return sbinary

    # perolehan binary image berdasarkan arah gradien
thresholding
    def dir_threshold(self, sobelx, sobely, thresh=(0,
np.pi/2)):

        abs_sobelx = np.abs(sobelx)
        abs_sobely = np.abs(sobely)
```

```
        grad_sobel = np.arctan2(abs_sobely, abs_sobelx)
        sbinary = np.zeros_like(grad_sobel)
        sbinary[(grad_sobel >= thresh[0]) & (grad_sobel <=
thresh[1])] = 1

        return sbinary

    # perolehan binary image berdasarkan penggabungan
thresholding
    def binary_extraction(self,image, ksize=3):

    # proses undistorsi
        color_bin = self.color_thresh(image,thresh=(90,
150))
        gray = cv2.cvtColor(image,
cv2.COLOR_RGB2GRAY)
        sobelx = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize)
        sobely = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize)
        gradx = self.abs_sobel_thresh(sobelx, thresh=(100,
190))
        grady = self.abs_sobel_thresh(sobely, thresh=(100,
190))
        mag_binary = self.mag_thresh(sobelx, sobely,
mag_thresh=(100, 190))
    # Pendektesian marka dengan Hough
        lines = cv2.HoughLinesP(mag_binary, rho=1,
theta=np.pi/180, threshold=30, minLineLength=40,
maxLineGap=100)
        left_slopes = []
        right_slopes = []
        avg_slope=0
        for line in lines:
            x1, y1, x2, y2 = line[0]
            cv2.line(blank_image,(x1,y1),(x2,y2),(0,0,255),2)
            angle = math.degrees(math.atan((y1-y2)/float(x1-
x2)))
            return avg_slope
```

```
        combined = np.zeros_like(gradx)
        #combined[(((gradx == 1) & (grady == 1)) |
((mag_binary == 1) & (dir_binary == 1))) | (color_bin==1)
] = 1
        combined[(((gradx == 1) & (grady == 1)) |
(mag_binary == 1)) | (color_bin==1) ] = 1
        #combined[(((gradx == 1) & (grady == 1)) |
(mag_binary == 1)) ] = 1

        return combined

    # perubahan perspektif
    def trans_per(self, image):

        image = self.binary_extraction(image)

        self.binary_image = image

        ysize = image.shape[0]
        xsize = image.shape[1]

        # mendefinisikan region of interest (roi)
        left_bottom = (xsize/10, ysize)
        apex_l = (xsize/2 - 2600/(self.look_ahead**2),  ysize
- self.look_ahead*275/30)
        apex_r = (xsize/2 + 2600/(self.look_ahead**2),  ysize
- self.look_ahead*275/30)
        right_bottom = (xsize - xsize/10, ysize)

        #  mendefinisikan  vertices  for  perspective
transformation
        src  =  np.array([[left_bottom],  [apex_l],  [apex_r],
[right_bottom]], dtype=np.float32)
        dst  =  np.float32([[xsize/3,ysize],[xsize/4.5,0],[xsize-
xsize/4.5,0],[xsize-xsize/3, ysize]])

        self.M = cv2.getPerspectiveTransform(src, dst)
```

```
        self.Minv = cv2.getPerspectiveTransform(dst, src)

        if len(image.shape) > 2:
            warped = cv2.warpPerspective(image, self.M,
image.shape[-2:None:-1], flags=cv2.INTER_LINEAR)
        else:
            warped = cv2.warpPerspective(image, self.M,
image.shape[-1:None:-1], flags=cv2.INTER_LINEAR)
        return warped

    # memunculkan masking pada tampilan windows
terhadap deteksi marka
    def window_mask(self, width, height, img_ref,
center,level):
        output = np.zeros_like(img_ref)
        output[int(img_ref.shape[0]-
(level+1)*height):int(img_ref.shape[0]-level*height), \
               max(0,int(center-
width/2)):min(int(center+width/2),img_ref.shape[1])] = 1
        return output

    # menentukan titik pusat / centroid pada posisi kiri dan
kanan marka
    def find_window_centroids(self, warped,
window_width, window_height, margin):

        window_centroids = []
        window = np.ones(window_width)

        # pertama-tama mencari 2 posisi awal pada kiri dan
kanan marka dengan menggunakan np.sum untuk
memperoleh perpotongan gambar vertikal

        # selanjutnya menggunakan np.convolve pada
gambar vertikal dengan template windows
```

```
        # jumlahkan seperempat gambar dari bawah untuk memperoleh potongan. dapat digunakan rasio yang berbeda

        l_sum = np.sum(warped[int(3*warped.shape[0]/4):,:int(warped.shape[1]/2)], axis=0)
        l_center = np.argmax(np.convolve(window,l_sum))-window_width/2
        r_sum = np.sum(warped[int(3*warped.shape[0]/4):,int(warped.shape[1]/2):], axis=0)
        r_center = np.argmax(np.convolve(window,r_sum))-window_width/2+int(warped.shape[1]/2)

        #penambahan pada layer pertama
        window_centroids.append((l_center,r_center))

        # mengoreksi setiap layer untuk mencari nilai maks lokasi
        for level in range(1,(int)(warped.shape[0]/window_height)):
            # mengkonvolusikan windows ke gambar perpotongan vertikal
            image_layer = np.sum(warped[int(warped.shape[0]-(level+1)*window_height):int(warped.shape[0]-level*window_height),:], axis=0)
            conv_signal = np.convolve(window, image_layer)
            # menentukan nilai centroid sebelah kiri dengan pusat kiri sebelumnya sebagai referensi
            # menentukan nilai offset dibawah karena referensi sinyal konvolusi berada sisi kiri window
            offset = window_width/2
            l_min_index = int(max(l_center+offset-margin,0))
            l_max_index = int(min(l_center+offset+margin,warped.shape[1]))
```

```
            l_center =
np.argmax(conv_signal[l_min_index:l_max_index])+l_m
in_index-offset
            # menentukan nilai centroid sebelah kiri dengan
pusat kiri sebelumnya sebagai referensi
            r_min_index = int(max(r_center+offset-margin,0))
            r_max_index =
int(min(r_center+offset+margin,warped.shape[1]))
            r_center =
np.argmax(conv_signal[r_min_index:r_max_index])+r_m
in_index-offset
            # tambahan pada layer
            window_centroids.append((l_center,r_center))

        return window_centroids

    # persamaan yang cocok pada marka kiri dan kanan
    def get_fit(self, image):

        # pengecekan jika marka terdeteksi pada iterasi
terakir. tetap mencari jika tidak terdeteksi
        if not self.detected:
            # window settings
            window_width = 40
            window_height = 40 # pemisahan gambar menjadi
9 vrtikal pada tinggi 720
            margin = 10 # berapa banyak slide sisi kiri dan
kanan untuk pencarian

            window_centroids =
self.find_window_centroids(image, window_width,
window_height, margin)

            # jika menemukan pusat dari tampilan
            if len(window_centroids) > 0:
```

```
            # point untuk menggambar seluruh sisi kiri dan
kanan tampilan
            l_points = np.zeros_like(image)
            r_points = np.zeros_like(image)

            # penggambaran tampilan pada setiap level
            for level in range(0,len(window_centroids)):
              # mask digunakan untuk menggambar area
yang terdeteksi
              l_mask                                    =
self.window_mask(window_width,window_height,image
,window_centroids[level][0],level)
              r_mask                                    =
self.window_mask(window_width,window_height,image
,window_centroids[level][1],level)
              #
              l_points[(image == 1) & (l_mask == 1) ] = 1
              r_points[(image == 1) & (r_mask == 1) ] = 1

            # pembuatan hasil dari imaging
            template_l = np.array(l_points*255,np.uint8) #
add left window pixels
            template_r = np.array(r_points*255,np.uint8) #
add right window pixels
            zero_channel = np.zeros_like(template_l) #
create a zero color channel
            left_right                                  =
np.array(cv2.merge((template_l,zero_channel,template_r)
),np.uint8) # make color image left and right lane

            # menentukan point pada polinomial fit
            self.allyl,self.allxl = l_points.nonzero()
            self.allyr,self.allxr = r_points.nonzero()


            # mengecek apakah garis marka terdeteksi
dengan benar
```

```
            if (len(self.allxl)>0) & (len(self.allxr)>0):
                try:
                    self.current_fit_l                    =
np.polyfit(self.allyl,self.allxl, 2)
                    self.current_fit_r                    =
np.polyfit(self.allyr,self.allxr, 2)
                    self.poly_warning = False
                except np.RankWarning:
                    self.poly_warning = True
                    pass

                # pengecekan marka
                if self.check_fit():
                    self.detected = True


                    if not self.detected_first:
                        self.best_fit_l = self.current_fit_l
                        self.best_fit_r = self.current_fit_r

                    else:
                        self.best_fit_l   =   self.best_fit_l*0.6   +
self.current_fit_l * 0.4
                        self.best_fit_r   =   self.best_fit_r*0.6   +
self.current_fit_r * 0.4


                    self.detected_first = True
                    self.bestxl = self.allxl
                    self.bestyl = self.allyl
                    self.bestxr = self.allxr
                    self.bestyr = self.allyr
                    self.left_right = left_right


                else:
                    self.detected = False
```

```
        # bila frame marka terdeteksi pada frame terakir, pencarian area frame yang tersedia
        else:
            non_zero_y, non_zero_x = image.nonzero()

            margin = 10 # pencarian margin area
            left_lane_points_indx = ((non_zero_x > (self.best_fit_l[0]*(non_zero_y**2) + self.best_fit_l[1]*non_zero_y + self.best_fit_l[2] - margin)) & (non_zero_x < (self.best_fit_l[0]*(non_zero_y**2) + self.best_fit_l[1]*non_zero_y + self.best_fit_l[2] + margin)))
            right_lane_points_indx = ((non_zero_x > (self.best_fit_r[0]*(non_zero_y**2) + self.best_fit_r[1]*non_zero_y + self.best_fit_r[2] - margin)) & (non_zero_x < (self.best_fit_r[0]*(non_zero_y**2) + self.best_fit_r[1]*non_zero_y + self.best_fit_r[2] + margin)))

            # ekstraksi piksel marka kiri
            self.allxl= non_zero_x[left_lane_points_indx]
            self.allyl= non_zero_y[left_lane_points_indx]


            # ekstraksi piksel marka kanan
            self.allxr= non_zero_x[right_lane_points_indx]
            self.allyr= non_zero_y[right_lane_points_indx]

            # jika garis marka ditemukan
            if (len(self.allxl)>0) & (len(self.allxr)>0):
                try:
                    self.current_fit_l = np.polyfit(self.allyl,self.allxl, 2)
                    self.current_fit_r = np.polyfit(self.allyr,self.allxr, 2)
```

```
            except np.RankWarning:
                self.poly_warning = True
                pass

            # mengecek apakah garis marka terdeteksi dengan benar
            if self.check_fit():
                # merata - rata hasil yang mendekati marka dengan nilai yang baru
                self.best_fit_l = self.best_fit_l*0.6 + self.current_fit_l * 0.4
                self.best_fit_r = self.best_fit_r*0.6 + self.current_fit_r * 0.4


                self.bestxl = self.allxl
                self.bestyl = self.allyl
                self.bestxr = self.allxr
                self.bestyr = self.allyr

                # membuat gambar pada hasil
                template_l = np.copy(image).astype(np.uint8)
                template_r = np.copy(image).astype(np.uint8)


template_l[non_zero_y[left_lane_points_indx],non_zero_x[left_lane_points_indx]] = 255 # add left window pixels

template_r[non_zero_y[right_lane_points_indx],non_zero_x[right_lane_points_indx]] = 255
                zero_channel = np.zeros_like(template_l) # create a zero color channel
                self.left_right = np.array(cv2.merge((template_l,zero_channel,template_r)),np.uint8) # penambahan warna marka kiri dan kanan
```

```
                # inisiasi ulang jika marka tidak terdeteksi
                else:
                    self.detected = False

    # mengecek apakah garis marka terdeteksi dengan benar
    def check_fit(self):
                ploty = np.linspace(0, self.im_shape[0]-1,
self.im_shape[0])
        left_fitx = self.current_fit_l[0]*ploty**2 +
self.current_fit_l[1]*ploty + self.current_fit_l[2]
        right_fitx = self.current_fit_r[0]*ploty**2 +
self.current_fit_r[1]*ploty + self.current_fit_r[2]

        # mencari nilai maks, min, dan rata" jarak antar marka
        max_dist  = np.amax(np.abs(right_fitx - left_fitx))
        min_dist  = np.amin(np.abs(right_fitx - left_fitx))
        mean_dist = np.mean(np.abs(right_fitx - left_fitx))

        if (max_dist > 250) | (np.abs(max_dist - mean_dist)>
100) | (np.abs(mean_dist - min_dist) > 100) |
(mean_dist<50) | self.poly_warning:
            return False
        else:
            return True

    # menentukan nilai radian/curvatur kelengkungan
marka
    def calculate_curvature_offset(self):

        if self.detected_first:

            y_eval = self.im_shape[0]

            # menentukan konversi pixel kedalam satuan centi
meter
            ym_per_pix = 0.2 # meter per pixel pada y dimensi
```

```
        xm_per_pix = 0.01 # meter per pixel pada x
dimensi

        # pembuatan persamaan x,y pada 2D
        try:
            left_fit_cr = np.polyfit(self.bestyl*ym_per_pix,
self.bestxl*xm_per_pix, 2)
            right_fit_cr                                    =
np.polyfit(self.bestyr*ym_per_pix,
self.bestxr*xm_per_pix, 2)
        except np.RankWarning:
            self.poly_warning = True
            pass


        if not self.poly_warning:
            # rumus perhitungan radius kelengkungan
            left_curverad          =          ((1          +
(2*left_fit_cr[0]*y_eval*ym_per_pix                        +
left_fit_cr[1])**2)**1.5) / np.absolute(2*left_fit_cr[0])
            right_curverad         =          ((1          +
(2*right_fit_cr[0]*y_eval*ym_per_pix                       +
right_fit_cr[1])**2)**1.5) / np.absolute(2*right_fit_cr[0])


            # perhitungan offset pada pusat jalan
            y_eval = y_eval*ym_per_pix
            midpoint_car = self.im_shape[1]/2.0
            midpoint_lane  =(right_fit_cr[0]*(y_eval**2)  +
right_fit_cr[1]*y_eval + right_fit_cr[2]) + \
                        (left_fit_cr[0]*(y_eval**2)        +
left_fit_cr[1]*y_eval + left_fit_cr[2])

            offset      =      midpoint_car*xm_per_pix     -
midpoint_lane/2
```

```
                if self.left_curverad == None:
                    self.left_curverad = left_curverad
                    self.right_curverad = right_curverad
                    self.offset = offset

                # rata-rata nilai offset kelengkungan
                else:
                    self.left_curverad = self.left_curverad * 0.8 +
left_curverad*0.2
                    self.right_curverad = self.right_curverad * 0.8
+ right_curverad*0.2
                    self.offset = self.offset * 0.9 + offset*0.1

    # hasil  pada sumber gambar
    def project_on_road_debug(self, image_input):
        image = image_input[self.remove_pixels:, :]
        image = self.trans_per(image)
        self.im_shape = image.shape
        self.get_fit(image)

        if self.detected_first & self.detected:
            # membuat fill image
            temp_filler                                          =
np.zeros((self.remove_pixels,self.im_shape[1])).astype(n
p.uint8)
            filler                                               =
np.dstack((temp_filler,temp_filler,temp_filler))

            # membuat suatu image untuk mengaktifkan garis
            warp_zero                                            =
np.zeros_like(image).astype(np.uint8)
            color_warp = np.dstack((warp_zero, warp_zero,
warp_zero))

            ploty = np.linspace(0, image_input.shape[0]-1,
image_input.shape[0] )
```

```
        left_fitx = self.best_fit_l[0]*ploty**2 +
self.best_fit_l[1]*ploty + self.best_fit_l[2]
        right_fitx = self.best_fit_r[0]*ploty**2 +
self.best_fit_r[1]*ploty + self.best_fit_r[2]

        cv2.fillPoly()
        pts_left =
np.array([np.transpose(np.vstack([left_fitx, ploty]))])
        pts_right =
np.array([np.flipud(np.transpose(np.vstack([right_fitx,
ploty])))])
        pts = np.hstack((pts_left, pts_right))
        cv2.fillPoly(color_warp, np.int_([pts]), (0,255, 0))

        newwarp = cv2.warpPerspective(color_warp,
self.Minv, color_warp.shape[-2:None:-1])
        left_right = cv2.warpPerspective(self.left_right,
self.Minv, color_warp.shape[-2:None:-1])
        # menggabungkan hasil imaging dengan gambar
yang asli
        left_right_fill = np.vstack((filler,left_right))
        result = cv2.addWeighted(left_right_fill,1,
image_input, 1, 0)
        result = cv2.addWeighted(result, 1,
np.vstack((filler,newwarp)), 0.3, 0)

        # cari nilai kelengkungan dan offset
        self.calculate_curvature_offset()

        # memploting tulisan string pada hasil gambar
        img_text = "radius kelengkungan: " +
str(round((self.left_curverad + self.right_curverad)/2,2)) +
' (cm)'
        status = "Satu Jalur!"
           # menentukan kontour gambar
        (cnts, _) = cv2.findContours(edged.copy(),
cv2.RETR_EXTERNAL,
```

```
                cv2.CHAIN_APPROX_SIMPLE)
                # looping kontour
        for c in cnts:
                # perhitungan kontur
                peri = cv2.arcLength(c, True)
                approx = cv2.approxPolyDP(c, 0.01 * peri,
True)

                # pembentukan persegi pada deteksi
                if len(approx) >= 4 and len(approx) <= 6:

                        # memasukkan float kotak box deteksi
                        (x, y, w, h) = cv2.boundingRect(approx)
                        aspectRatio = w / float(h)

                        # perhitungan area kontur
                        area = cv2.contourArea(c)
                        hullArea                        =
cv2.contourArea(cv2.convexHull(c))
                        solidity = area / float(hullArea)
                        keepDims = w > 25 and h > 25
                        keepSolidity = solidity > 0.9
                        keepAspectRatio = aspectRatio >= 0.8
and aspectRatio <= 1.2

                        # apakah kontour berhasil
                        if keepDims and keepSolidity and
keepAspectRatio:
                                # jika pattern terdeteksi
                                cv2.drawContours(frame, [approx],
-1, (0, 0, 255), 4)
                                status = "Boleh menyalip!"

                                # perhitungan titik tengah kontour
dan memberikan efek crosshair
                                M = cv2.moments(approx)
```

```
                    (cX, cY) = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
                    (startX, endX) = (int(cX - (w * 0.15)), int(cX + (w * 0.15)))
                    (startY, endY) = (int(cY - (h * 0.15)), int(cY + (h * 0.15)))
                    cv2.line(frame, (startX, cY), (endX, cY), (0, 0, 255), 3)
                    cv2.line(frame, (cX, startY), (cX, endY), (0, 0, 255), 3)

# memberi status teks pada gambar
        cv2.putText(frame, status, (20, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(0, 0, 255), 2)

        if self.offset< 0:
          img_text2 = "Posisi Kendaraan: " + str(round(np.abs(self.offset+2),2)) + ' (cm) kanan pada tengah'+ str(round(np.abs(9 - self.offset+2),2)) + ' (cm) pinggir kanan'

        else:
          img_text2 = "Posisi Kendaraan: " + str(round(np.abs(self.offset+2),2)) + ' (cm) kiri pada tengah'+ str(round(np.abs(9 - self.offset+2),2)) + ' (cm) pinggir kiri'

        small = cv2.resize(left_right_fill, (0,0), fx=0.5, fy=0.5)
        small2 = cv2.resize(np.vstack((filler,self.left_right)), (0,0), fx=0.5, fy=0.5)

        result2 = cv2.resize(np.hstack((result, np.vstack((small2,small)))), (0,0), fx=self.enlarge, fy=self.enlarge)
```

```
            cv2.putText(result2,img_text, (15,15),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255,255,255),1)
            cv2.putText(result2,img_text2,(15,40),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255,255,255),1)
            cv2.putText(result2,status,(30,90),
cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0,0,255),1)

            return result2

        # jika marka tidak terdeteksi pada pengambilan
sumber
        else:
            return_image =
cv2.resize(np.hstack((image_input,cv2.resize(np.zeros_li
ke(image_input),(0,0), fx=0.5, fy=1.0))),(0,0),
fx=self.enlarge, fy=self.enlarge)
            return return_image

    # hasil percobaan pada sumber gambar
    def project_on_road(self, image_input):
        image = image_input[self.remove_pixels:, :]
        image = self.trans_per(image)
        self.im_shape = image.shape
        self.get_fit(image)

        if self.detected_first & self.detected:
            # membuat fill image
            temp_filler =
np.zeros((self.remove_pixels,self.im_shape[1])).astype(n
p.uint8)
            filler =
np.dstack((temp_filler,temp_filler,temp_filler))

            # membuat suatu image untuk mengaktifkan garis
            warp_zero =
np.zeros_like(image).astype(np.uint8)
```

```
        color_warp = np.dstack((warp_zero, warp_zero,
warp_zero))

        ploty = np.linspace(0, image_input.shape[0]-1,
image_input.shape[0] )
        left_fitx = self.best_fit_l[0]*ploty**2 +
self.best_fit_l[1]*ploty + self.best_fit_l[2]
        right_fitx = self.best_fit_r[0]*ploty**2 +
self.best_fit_r[1]*ploty + self.best_fit_r[2]

cv2.fillPoly()
        pts_left =
np.array([np.transpose(np.vstack([left_fitx, ploty]))])
        pts_right =
np.array([np.flipud(np.transpose(np.vstack([right_fitx,
ploty])))])
        pts = np.hstack((pts_left, pts_right))

        # menggambar garis marka pada gambar mentah
        cv2.fillPoly(color_warp, np.int_([pts]), (0,255, 0))

        newwarp = cv2.warpPerspective(color_warp,
self.Minv, color_warp.shape[-2:None:-1])
        left_right = cv2.warpPerspective(self.left_right,
self.Minv, color_warp.shape[-2:None:-1])

        # menggabungkan hasil imaging dengan gambar
yang asli
        left_right_fill = np.vstack((filler,left_right))
        result = cv2.addWeighted(left_right_fill,1,
image_input, 1, 0)
        result = cv2.addWeighted(result, 1,
np.vstack((filler,newwarp)), 0.3, 0)

        # cari nilai kelengkungan dan offset
        self.calculate_curvature_offset()
```

```
            # memploting tulisan string pada hasil gambar
            img_text = "radius kelengkungan: " +
str(round((self.left_curverad + self.right_curverad)/2,2)) +
' (cm)'

            if self.offset< 0:
               img_text2 = "Posisi Kendaraan: " +
str(round(np.abs(self.offset),2)) + ' (cm) kiri pada tengah'
            else:
               img_text2 = "Posisi Kendaraan: " +
str(round(np.abs(self.offset),2)) + ' (cm) kanan pada
tengah'

            result2 = cv2.resize(result, (0,0), fx=self.enlarge,
fy=self.enlarge)

            cv2.putText(result2,img_text,          (15,15),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255,255,255),1)
            cv2.putText(result2,img_text2,(15,40),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255,255,255),1)

            return result2

         # jika marka tidak terdeteksi pada pengambilan
sumber
         else:
            return                cv2.resize(image_input,(0,0),
fx=self.enlarge, fy=self.enlarge)
lines = Lines()
lines.look_ahead = 10
lines.remove_pixels = 100
lines.enlarge = 2.25

# mengambil frame dari kamera
for frame in camera.capture_continuous(rawCapture,
format="bgr", use_video_port=True):
```

```
        image = frame.array

        # menampilkan windows frame gambar

        cv2.imshow("Deteksi Marka Jalan",
lines.project_on_road_debug(image))
        key = cv2.waitKey(1) & 0xFF

        # merefresh frame gambar
        rawCapture.truncate()
        rawCapture.seek(0)

        # tombol 'q' untuk menghentikan loop
        if key == ord("q"):
                break
```

BIODATA PENULIS

Penulis bernama lengkap Tarezqi Mochtar Rohma yang akrab disapa Rezqi. Penulis merupakan anak ke-satu dari dua bersaudara, terlahir di kota Madiun pada tanggal 22 April 1996. Penulis menempuh pendidikan di MI ISLAMIYAH Madiun lulus tahun 2008, SMPN 3 Madiun lulus tahun 2011, dan SMAN 2 Madiun lulus tahun 2014. Pendidikan sarjana ditempuh di Jurusan Teknik Fisika ITS melalui jalur SNMPTN 2014. Selama aktif menjadi mahasiswa, penulis bergabung dalam organisasi akademik sebagai Asisten Laboratorium Kalibrasi dan Pengukuran Fisis2015 – 2018. Penulis mendapat pengalaman *Internship Program* selama 1 bulan di PT. YTL Paiton, Jawa Timur pada bidang *Electrical Control and Instrumentation*. Selain aktif dalam organisasi akademik , penulis juga aktif dalam kegiatan organisasi diluar akademik, yaitu sebagai Staff Unit Eksternal Kegiatan Mahasiswa Betako Merpati Putih pada 2015 – 2017. Bidang minat penulis dalam mengerjakan tugas akhir adalah Rekayasa Fotonika. Penulis dapat dihubungi di email *mochta03@gmail.com.*